UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

# WEPLANN: APLICACIÓ PER AUTORGANITZAR-SE

Autor: Raül Gómez Buisan

Director: Prof. Jordi José Bazán
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 27 de juny de 2018

# Abstract

This is a project in which an Android application has been developed with the aim of offering to users some tools to organize their time and plan tasks, routines or projects on a personal or groupal way. The main idea is that each user could add plans and, at the same time, other users can participate in the projects.

To achieve the objective, several technologies have been used. Some of them are typical Android, but some third-party libraries and tools have also been used. It has been used Kotlin language, which is supported by Android, for the entire application and Cloud Firestore for the database. In the last part of the project tests have been performed to several users with different characteristics to evaluate the app user interface.

At the end of the project, the initial planning was highly valued and the objectives that had been planned have been achieved.



# Resum

Aquest és un projecte en el que s'ha desenvolupat una aplicació Android amb la intenció d'oferir als usuaris varies eines per poder organitzar-se el temps i planificar-se tasques, rutines o projectes a nivell individual o grupal. La idea principal és que cadascun dels usuaris pugui afegir plans i alhora pugui fer particip dels projectes altres usuaris.

Per aconseguir l'objectiu esmentat s'han fet servir diverses tecnologies. Algunes són pròpies de les aplicacions d'Android però també s'han usat vàries llibreries i eines de tercers. Destaca l'ús del llenguatge Kotlin, que té el suport d'Android, per a tota l'aplicació i Cloud Firestore per a la base de dades. Hi ha una última part del projecte on s'han efectuat tests a varis usuaris amb característiques diferents per valorar la interfície d'usuari de l'aplicació.

Al acabar el projecte s'ha valorat molt positivament la planificació inicial i es pot apreciar com s'han aconseguit els objectius que s'havien planejat.

# Agraïments

Vull agrair a tots els que m'han acompanyat en aquest llarg camí,
vull dir-lis que sense ells això no hagués estat possible.
Avis, pares, germans, amics i professors
hi ha una part de cadascun de vosaltres en aquest projecte.
Moltes gràcies per aquest viatge.

# Índex

# 1 Introducció

## 1.1 El projecte

El projecte consisteix en crear una aplicació, per Android, capaç de permetre organitzar-se tant al dia a dia com a l'hora de realitzar projectes. La idea seria programar una app amb una interfície agradable per l'usuari i amb un funcionament fàcil d'entendre que ajudés als usuaris a establir fites i definir tasques per poder escollir en què desitgen emprar el temps.

Tot i que l'objectiu del projecte sigui un molt concret, com és crear una app, hi ha un seguit de tasques subsidiàries que són necessàries. Una d'aquestes és l'obligació d'adquirir nous coneixements per poder implementar diferents funcionalitats i, d'altra banda, la necessitat de planificar el temps i els elements que es desitgen implementar dintre del marc temporal i de treball definit pel TFG.

## 1.2 Motivació

Es podrien haver escollit diferents modalitats de projectes d'entre les diferents tecnologies apreses durant els estudis, però s'ha decidit realitzar una aplicació per mòbil. Els motius principals per aquesta elecció han estat que per una cantó hi ha força documentació a Internet, facilitant la solució d'errors, i per l'altre cantó la possibilitat d'arribar a un públic molt més ampli, inclosos als més joves, degut a l'ús freqüent de les tecnologies per gran part de la població:

> 'El uso generalizado de las nuevas tecnologías de la información y la comunicación (TIC) ha transformado las relaciones entre las personas en el mundo actual, lo que es especialmente relevante en el caso de los más jóvenes. La navegación por Internet, el uso de redes sociales virtuales, los videojuegos y el teléfono móvil han supuesto un cambio radical en las formas de comunicarse para la mayoría de ellos.' [1]

Inicialment, hi havia una altre idea de TFG però com es va canviar de professor també es va canviar el tema del treball final. Cal dir que realitzar una app amb aquestes característiques és una idea que venia de lluny, ja que quan s'ha de fer qualsevol rutina o projecte, ja sigui individual o grupal, les eines que existeixen no acaben d'encaixar amb les necessitats dels projectes. Hi ha varis problemes en les apps existents que més endavant s'analitzaran, e.g. cap té una interfície adaptable a projectes de diferents característiques i fàcil d'entendre per als diferents perfils d'usuaris. Així doncs, es pot dir que, aquest projecte sorgeix d'un problema individual que moltes persones es troben al seu dia a dia i s'espera poder aportar una alternativa que tracti aquests problemes.

En aquest treball s'han intentat plasmar diferents idees apreses en diferents assignatures de les que destaquen: *Factors Humans i Computació* i *Enginyeria del*

*Software*. Per una banda es volia definir una aplicació que tingués com a objectiu ser usable per usuaris amb diferents backgrounds. Per altra banda es pretenia que integrés elements de la metodologia **Agile** per a que l'usuari pogués distribuir-se el temps segons els seus interessos tot oferint la possibilitat d'adequar-se al màxim a tota mena d'exigències organitzatives. Es tracta doncs d'un intent d'universalitzar la planificació del temps en les vides de qualsevol usuari d'acord als seus propis objectius. Tot això, sense perdre el nord i tenint en compte alguns principis de la metodologia **Agile** entre els que destacaria:

> 'La simplicitat, l'art de maximitzar la quantitat de feina que no es fa, és essencial.'[2]

Tot i que puguin semblar uns grans objectius i motivacions per a un TFG, en aquest treball es pretén implementar una versió inicial, *beta*, que incorpori les principals funcionalitats a nivell individual i quedi preparat per a l'organització de grups tot incloent algunes funcionalitats de caràcter embrionari. Deixant preparat el camí per a una posterior implementació, que sigui capaç d'assumir un funcionament coherent entre l'ús individual i l'ús grupal de l'aplicació, més profunda i amb un conjunt d'eines per als grups que no són l'objectiu d'aquest treball.

# 2 Objectius

Es plantegen diferents objectius, alguns més grans que altres, que es podríen resumir en:

- Crear una app que permeti definir tasques
- Crear una app que permeti definir projectes
- Crear una app que permeti definir rutines
- Crear una app que implementi un calendari
- Crear una app que permeti el treball en grups
- Realitzar un testeig amb usuaris

Els anteriors són el conjunt d'objectius finals lligats a la nostre aplicació que marcaran la direcció del treball. A part, hi ha alguns objectius que tot i que estan fora de l'abast d'aquest projecte defineixen algunes línies a l'hora d'implementar l'aplicatiu. Aquestes intencions futures es podrien resumir en dos: per un costat poder convertir aquesta app a altres plataformes com IOS o una versió Web; i per l'altre, que tingui una alta modularitat que permeti afegir o modificar eines fàcilment.

## 2.1 Filosofia d'aplicació

'Comunicació, redistribució i cooperació a nivell grupal.'

'Creació, planificació i superació a nivell personal.'

Es busca oferir a l'usuari una nova forma d'organitzar-se segons projectes, però també per tasques eventuals o rutines, associant diferents tasques a un únic projecte. Alhora també es poden associar tant tasques, rutines com projectes a una o varies categories que un desitgi. Per a poder cooperar entre els diferents usuaris s'ofereix la possibilitat d'afegir altres usuaris a un projecte. Alhora es pretén que l'aplicació en un futur permetí comunicar-se de la forma més idònia als membres d'un projecte i redistribuir el treball entre els diferents usuaris.

No es vol arribar a ser cap nova xarxa social que absorbeixi el temps dels usuaris, ans el contrari. L'ideal que s'intenta estendre és el d'un ús responsable de les aplicacions, Internet, i en general de la tecnologia; quelcom que cap de les noves grans xarxes socials com *Facebook, Instagram, Twitter* o altres estan promovent. Deixant clar que es vol un ús de la tecnologia existent com a eina i no com a substitutiu de les vides de la gent. Fugint del que ha donat lloc, entre altres situacions, a l'*assumpte de Facebook*[3], que és la punta de l'iceberg entre el conjunt de problemes associats al traspàs que s'ha fet de la vida material i les relacions personals a les xarxes socials. Un traspàs que ha provocat que aquestes xarxes socials siguin

les fonts de satisfacció, informació, comunicació, entreteniment, etc; de moltes persones. On ni la informació verídica i contrastada ni les relacions sanes ni moltes altres qüestions formen part de les prioritats dels usuaris ni tampoc dels promotors d'aquestes xarxes socials.[4][5]

Està clar que no hi ha gaires possibilitats d'arribar a conscienciar a molts membres d'aquestes xarxes, però dintre del marge d'acció que dóna aquest projecte es farà el que es pugui. Es vol promoure una alternativa a l'emergent problema social associat a l'alt nivell de dependència, sobretot en els més joves, de les xarxes socials que està provocant greus problemes en el comportament i en les relacions d'aquesta *societat 2.0*. Una problemàtica que no només està provocant problemes en els més joves, sinó també en adults, de la mà d'escarnis públics que ensorren primer la reputació a les xarxes i tot seguit anul·len a la persona de la seva vida quotidiana.[6]

Les anteriors situacions es volen evitar fomentant la superació i la creativitat a nivell individual i la cooperació a nivell grupal per intentar evitar caure en la competitivitat, el descrèdit i el narcisisme predominants a les xarxes socials existents. Un plantejament que pot semblar un brindis al cel perquè no només depèn de l'aplicació, sinó també de l'ús dels usuaris. Tot i això, si es pensa en el funcionament i la utilitat de l'aplicació es pot apreciar com encaixen bastant bé filosofia i aplicació. Donat que l'app és una eina, en cap cas un substitutiu de la vida per a les persones que la usin, enfocada a empènyer aquestes persones a una activitat o un altre a la seva vida. Buscar motivar a les persones a perseguir els seus objectius tant a nivell individual, si es tracta d'objectius o tasques personals, com a nivell col·lectiu. Aspirant a desencadenar una evolució en l'ús de la tecnologia, des d'una perspectiva que està reemplaçant la vida material per una virtual, cap a unes aplicacions amb l'objectiu de potenciar la cooperació entre persones i la superació com a individus.

# 3 Planificació

## 3.1 Prèvia

La planificació prèvia va fer-se a principi de curs. Es va plantejar amb l'anterior professor que el primer quadrimestre s'estaria buscant documentació i formant-se sobre el tema; i el segon quadrimestre es desenvoluparia l'aplicatiu, que anava a ser una aplicació amb prolog encarada a diagnosticar la dislèxia.

Es van llegir articles que tractessin sobre el tema i contactant amb entitats que estaven abordant la temàtica. Finalment va haver-hi un projecte en el qual es va estar apunt de començar a treballar per fer TFG i ajudar-los a ells, *Projecte Binding*. D'altra banda, també s'estava estudiant el llenguatge de programació *Prolog*, amb el llibre *The Art of Prolog* de Leon Sterling.

Al començar el segon quadrimestre el professor que s'encarregava dirigir el TFG ara ja no podia i es va haver de canviar de tutor i alhora canviar la temàtica del TFG. Es va canviar la temàtica fruit de la desmotivació que s'havia generat arrel del que havia succeït amb el professor i per altra banda van sorgir moltes ganes de crear una aplicació i poder aprofitar-la després com a projecte personal tot utilitzant les tecnologies tant a nivell de llenguatges com de qualsevol altre mena més potents.

Així doncs, tot i que semblava que la planificació prèvia anava per bon camí no va acabar servint per a res.

## 3.2 Inicial

Un cop començat el segon quadrimestre es va assignar un tutor per al treball, Jordi José Bazan, amb el que es va concretar un reunió el més ràpid possible. En aquesta primera reunió va presentar-se-li la situació del projecte: s'havia d'iniciar des de zero un nou TFG del qual no s'havien delimitat els objectius. Tot i això hi havia molta motivació al darrere. Es sabia quina mena d'aplicació es volia fer, per Android, però no quines tecnologies s'usarien ni s'havien definit la majoria de funcionalitats de l'app, només s'havien aclarit les principals.

A la sortida d'aquesta reunió es va confirmar que s'havia de planificar els temps i sense fer volar coloms assumir fins on arribaria la implementació. Per això es va decidir que entre el març i l'abril s'hauria d'escollir quines eines s'utilitzarien i s'aprendria a fer-les sevir si no es coneixia com funcionaven. En els dos primers mesos es volia tenir l'aplicació acabada i l'esquelet de la memòria també. Primer, era necessari tenir la idea del projecte i objectius tancats. Després definir com desenvolupar l'aplicatiu i finalment tenir tancat quines tecnologies es farien servir. Finalment desenvolupar l'app i la memòria a partir de les línies que s'haguessin marcat anteriorment.

Arribats a aquest punt hi havia a grans trets definida una planificació des de l'inici fins a la fi del projecte. Els primers dos mesos es dedicaran a escollir les tecnologies i formar-se al voltant d'aquestes; també s'iniciarà el projecte, a nivell de programació, i s'intentarà tenir l'esquelet de la memòria. L'últim mes tocaria acabar l'aplicació, acabar la memòria i si donés temps realitzar un testeig amb usuaris de l'aplicació. Hi haurà una dedicació a temps complet durant abril, maig i juny. El motiu principal d'aquesta planificació ha estat la situació laboral, ja que en aquell moment al estar treballant no hi havia temps suficient per dedicar-se a temps complet al treball. A l'abril s'acabaven les pràctiques i es tenia la idea de dedicar-s'hi a temps complet.

## 3.3 A mitjans

A principis de maig quedaven dos mesos per a l'entrega de la memòria i el codi font. Mirant la planificació que s'havia fet es podia afirmar que s'estaven seguint les línies dintre que s'havien plantejat. Ara tocava la part més dura, s'havia d'acabar tant l'aplicació com la memòria i hi havia bastantes ganes d'incloure en el treball un testeig de l'app per part d'usuaris amb diferents backgrounds. Un cop arribats a aquest punt era important posar els peus a terra veure si s'estava anant de les mans el projecte, si calia treure-li funcionalitats o si per contra es podia continuar pel camí marcat.

El projecte estava ja avançat però ni molt menys acabat. Hi havia dos cartes però que jugaven a favor: per una banda hi havia dedicació completa a nivell temporal; i per l'altre molta motivació amb el projecte. Per contra, es podia anar de les mans: malgastar temps; que implementant algunes funcionalitats aquestes donessin més problemes dels esperats; o que sorgissin qualsevol altre mena de problemes que impedissin acabar el treball en els terminis desitjats. Així doncs, quedava clar que aquesta planificació era la més decisiva i era necessari estar a l'aguait i valorar les diferents línies de treball.

Es va decidir dividir el treball en tres apartats: **Aplicació**, **Memòria** i **Testeig**. El bloc de l'*App* i la *Memòria* serien obligatoris però en canvi el bloc de *Test* seria opcional i només es desenvoluparia si quan quedés un mes per l'entrega l'apartat *App* estigués enllestit.

L'aplicació tindria també una part opcional i s'implementaria si dona temps. Es va dividir l'App en tres parts: la **Lógica interna**, la **Individual** i **Grupal**. La *Lógica* contindria, segons s'hagués definit anteriorment, la lògica interna del funcionament de l'app incloent tant la distribució de tasques; l'agrupació i categorització d'aquestes; quines limitacions hi haurien; etc. Després l'apartat *Individual* seria on es treballaria per definir la interfície amb l'usuari a nivell individual i com aquesta interactua amb la *Logica interna*. Finalment hi haurà una part opcional que seria l'apartat *Grupal* que tractaria d'implementar un sistema que interactuant tant amb la lògica interna com individual concedís la possibilitat d'organitzar-se en diferents

grups tant per aconseguir uns objectius, amb un projecte, com per coordinar-se entre família/amics com per qualsevol altre motiu.

Dintre de la mateixa aplicació hi ha dos elements que tot i que no són menys importants s'han valorat per planificar-se. Per una banda el patró de disseny a seguir a l'hora de programar i per l'altre el menú de l'aplicació.

A continuació tenim la *Memòria* que es va decidir no dividir-la en parts perquè es tracta d'una tasca continuada i el procés ha estat lineal i no per parts. Així doncs, quan es treballés en la *Memòria* es faria intentant plasmar el que s'estigués fent, la informació que cercada o el procés que s'havia seguit. Així doncs, no tenia sentit haver-la dividit en diferents parts.

El *Test*, que es va decidir que només existiria si l'*App* s'havia acabat, tindria dues parts: la **Preparació** i l'**Execució** dels mateixos test pels usuaris. La idea seria que l'aplicació es testés durant dues setmanes més o menys amb un seguit de premisses i quan s'acabés el temps de prova es poguessin realitzar uns qüestionaris sobre la mateixa.

Tot i que es recomana que la memòria s'escrigui al final, alguns punts els s'han redactat abans d'acabar el TFG. Com és el cas de la comparativa de tecnologies o els apartats relacionats amb la planificació o la cerca de certa informació.

Destaca l'ús de el sistema de treball *Pomodoro* per a distribuir el temps. La distribució de tasques anterior ens permet implementar *Pomodoro* correctament per a cadascun dels apartats o subapartats.

# 4 Tecnologies

A continuació s'expliquen les diferents tecnologies usades en el projecte. Per una banda estarà: el sistema operatiu per al que anirà destinada; l'entorn de treball, IDE; el llenguatge que s'ha fet servir, *Kotlin*; i els components principals que formen part de les apps. D'altra banda hi ha el conjunt d'elements que no son propis de qualsevol aplicació Android sinó que són complements: com ara *Cloud Firestore*, per a la base de dades; o *MVP + Interactor* per a l'estructura de classes; *MaterialDrawer* de Mikepenz per al menú; etc.

Alhora es mostraran les comparatives que s'han realitzat per poder escollir una tecnologia o una altre. Així doncs, es podrà apreciar quines han estat les motivacions que han portat a triar entre les diferents possibilitats que s'oferien a l'hora de crear l'app.

## 4.1 Android

S'ha decidit apostar per Android per raons diverses. Per un costat, perquè ens permet arribar un nombre molt major de persones degut a l'ús majoritari d'aquest sistema operatiu per part de la majoria d'usuaris:

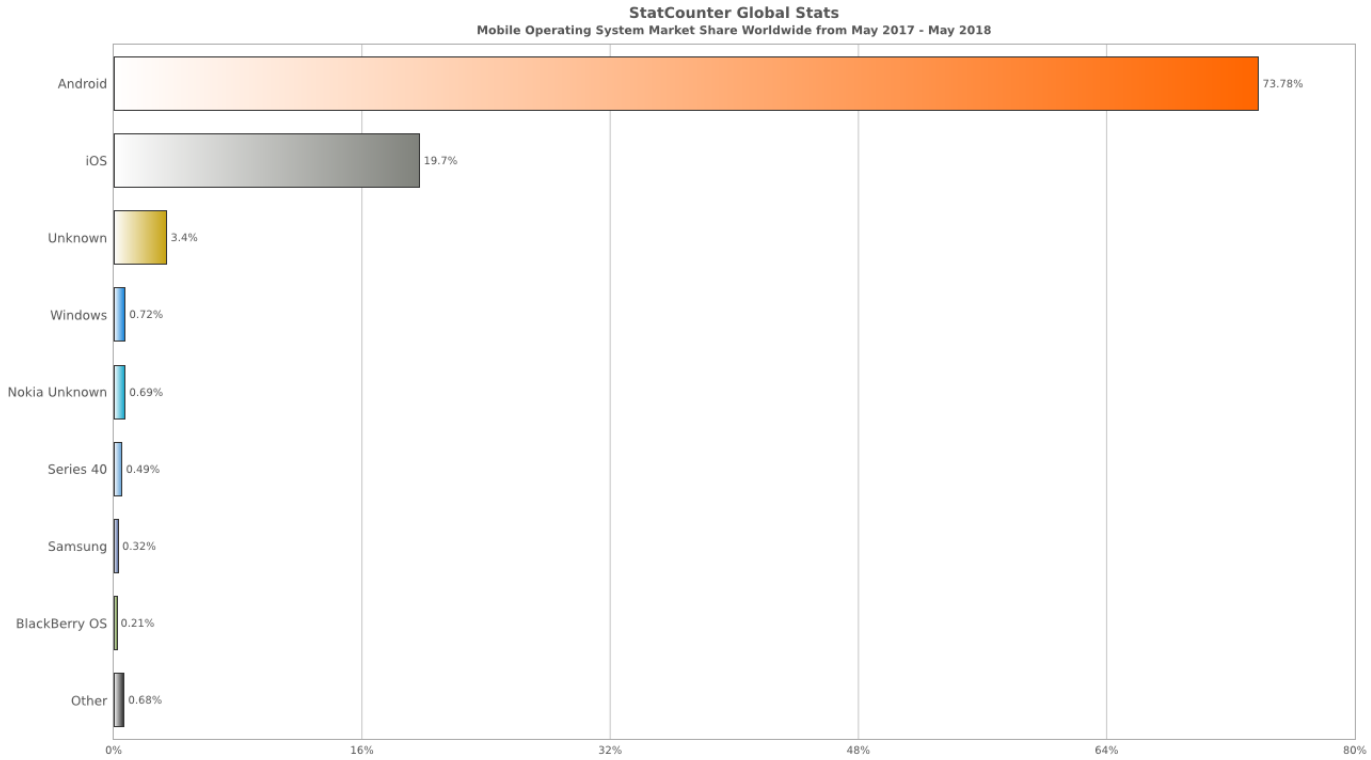


Figura 1: Percentatges d'us del Maig 2017 al Maig 2018[7]

Per l'altre, per la facilitat a l'hora implementar una app android després d'haver aprés a grans trets el procés de creació d'una aplicació a l'assignatura de *Projecte*

*Integrat de Software*. El present treball es diferencia d'aquesta assignatura esmentada per les noves eines que utilitza: tant per la base de dades, com pel llenguatge de programació i sobretot per l'objectiu, que en aquest cas no és fer cap joc, sinó una aplicació d'ús generalitzat.

Les anteriors raons han estat motius de pes per treballar sobre Android, una bona opció que permet fer un llançament inicial de l'aplicació i veure l'acceptació que pot tenir entre els diferents usuaris. Tanmateix, després d'obtenir un *feedback* de la comunitat Android es tindria informació a l'hora de decidir si es llença en un altre mercat com el d'IOS o en format Web i millorar els errors reportats a partir de la difusió que s'hagués fet fins al moment entre els dispositius Android.

### 4.1.1 Llenguatge: Kotlin, perquè?

Per a decidir quin llenguatge usar, tot i que s'hagués establert per a quin SO mòbil estaria enfocada l'aplicació, calia establir quin llenguatge de programació s'usaria. Al mercat hi ha moltes opcions, cadascuna amb els seus pros i els seus contres, és per això que per tal de valorar quina seria la més viable es van definir uns paràmetres. Aquests paràmetres provenen tant dels objectius inicials com de les intencions futures.

Després d'una llarga cerca mitjançant xerrades, articles i videocrítiques s'ha decidit usar el llenguatge *Kotlin*. Una decisió que en cap cas ha estat presa a l'atzar, sinó que ha estat fruit d'una comparativa exhaustiva entre les diferents opcions de desenvolupament.

Entre les diferents possibilitats s'ha fet una clara distinció entre les que oferien poder en un futur implementar fàcilment aplicacions en altres plataformes i les que no. Així doncs, s'han valorat les opcions següents: Flutter, Xamarin, Ionic, Reactive Native, Java and Kotlin.

Per una banda hi ha el desenvolupament multiplataforma, on els principals problemes són que: com no es tracta de les eines oficials depens dels desenvolupadors d'aquestes eines, menys en el cas de Flutter; desenvolupen interfícies alternatives a la oficial de forma que l'usuari pot notar aquesta diferencia; i sobretot on més problemes es poden trobar és a l'hora d'usar certes llibreries, podent existir la possibilitat d'haver de desenvolupar autònomament certes funcionalitats. Existia un problema afegit, en el cas de no saber cap dels llenguatges multiplataforma usats, com era el cas, s'havien d'haver après. Un avantatge principal era que es podria reutilitzar una gran part del codi i així estalviar-se haver de reprogramar-lo per cada plataforma. Tot i això hi ha certs elements que requeririen d'implementacions paral·leles en codi nadiu associades a cada plataforma on es volgués desenvolupar l'aplicatiu, e.g. la gestió de la càmera del mòbil. Dintre de les diverses opcions la que menys cridava l'atenció era Ionic, perquè no usa elements nadius en absolut. En lloc d'això, només mostra una pàgina web escrita en HTML que imita el disseny de reproductors nadius, amb tots els desavantatges que això comporta respecte un

desenvolupament nadiu. En el cas de Reactive Native tot i que és una molt bona opció té un rendiment pitjor i no ofereix un aspecte natiu a la vostra aplicació de forma que no es veu com la resta d'aplicacions que l'usuari usa. Xamarin era una molt bona opció però el desconeixement del llenguatge C#, i la possibilitat de que sorgissin dificultats per aquest factor, ha estat el principal limitador, ja que si no fos per això es tracta d'un desenvolupament que al final del projecte es troba molt proper al que hagués pogut fer-se amb un desenvolupament nadiu.[8] Finalment cal destacar que va existir seriosament el plantejament de programar l'aplicatiu amb Flutter, però al estar encara en fase beta, amb tot el que això comporta, va frenar aquesta idea.[9] Si encara hi hagués suport per RoboVM, al usar Java, hagués estat probablement l'opció escollida.

D'altra banda, es va valorar entre usar Java o Kotlin com a llenguatges nadius. Tots dos són llenguatges reconeguts oficialment, però aquest últim fa poc temps que ha estat reconegut per Android[10]. En aquest punt com cap de les opcions de multiplataforma existents ha tingut la capacitat de desmarcar-se de la resta. Això a provocat que es vegi una alta viabilitat a l'hora d'usar Java o Kotlin però hi ha hagut varis factors que han decantat la balança. El primer ha estat que és completament interoperatiu amb Java. El segon ha estat que és molt fàcil d'aprendre i obre tot un nou marc de possibilitats a l'hora de programar que no es tenia amb Java com ara la programació funcional, lambda's, data classes, permet crear un Domain-Specific Language per al projecte i moltes altres funcionalitats avançades. Permet, al poder treballar amb variables no nules, oblidar-se dels famosos *"null pointer exception"*. El tercer, i no per això menys important, és que estan desenvolupant una versió de Kotlin anomenada Kotlin/Native que permetrà la interoperabilitat entre diferents plataformes. Encara està en versió beta, però ja hi ha varies aplicacions multiplataforma desenvolupades amb Kotlin.[11] Per tot això, enfront les diferents opcions multiplataforma i java s'ha escollit Kotlin.

### 4.1.2 IDE: Android Studio

Un cop decidit com es desenvoluparia l'app s'havia d'escollir un IDE en el que implementar l'aplicació que facilités la feina al màxim. En aquest punt no es van plantejar tantes opcions com a l'hora d'escollir un desenvolupament multiplataforma o no i l'únic debat estava entre programar-la amb Android Studio o amb IntelliJ IDEA.

Un cop s'havia arribat fins aquí el que va marcar la diferència va ser que ja s'havia usat Android Studio i que aquest actualment ja estigués donant suport a Kotlin. Android Studio no ha donat cap problema en anteriors projectes i és per això que era des d'un principi la primera opció si l'aplicació es programava en nadiu. Està clar que IntelliJ IDEA també donava suport a Kotlin, però no s'havia treballat amb aquest IDE. A més, com no hi ha massa dificultat a l'hora de passar un projecte d'un IDE a l'altre, no s'ha vist cap problema en començar usant el IDE que més comoditat donés per desenvolupar el project. Definitivament es va optar per usar Android Studio com a IDE.

## 4.2 Database: Cloud Firestore

Una vegada decidit com s'implementarà l'aplicació i tenint en compte els objectius que es pretenen assolir s'havia de decidir si s'usaria base de dades i en cas que s'usés, escollir quin sistema de base de dades seria.

Per poder prendre la decisió va ser necessari establir la importància dels diferents objectius. Clarament la nostra aplicació requeria guardar informació de l'usuari i això es podia fer de dues maneres: localment, de forma que no calia cap base de dades; o guardar la informació al servidor, de forma que es necessitaria una base de dades que s'hauria de gestionar. Per determinar la millor opció es va donar importància a les futures intencions que es tenien amb l'aplicatiu associades a poder compartir tasques amb altres usuaris, a poder realitzar projectes col·lectius, etc. D'aquesta manera es va veure com clarament l'aplicació estava obligada a tenir una base de dades que pogués guardar la informació col·lectiva i alhora accedir-hi de diverses maneres. Això solucionava el problema individual lligat a guardar les dades relacionades només amb un mateix perquè es guardarien al núvol també.

Un cop es va arribar al punt d'acceptar que per aquest projecte es necessitava una base de dades es varen plantejar dues opcions: implementar una base de dades SQL des de cero amb totes les dificultats associades i entenent que seria força limitada degut a que no es tenia un coneixement profund sobre com implementar bases de dades; o usar les possibilitats que ofereix Google mitjançant Firebase. Sense dubtar es va escollir la segona opció.

Un cop decidit l'ús de Firebase, recomanat tant per companys d'universitat com pel tutor, per implementar la base de dades va aparèixer un nou sistema de base de dades de Google anomenat Cloud Firestore. Al començar a buscar per la xarxa els pros i contres respecte Realtime Database es va trobar que el principal contra era que estava en fase beta i una altre crítica era que Realtime Database té molt baixa latència i funciona millor per apps que necessiten una sincronització freqüent d'estats, e.g. jocs multijugador. D'altra banda, els avantatges que ofereix Cloud Firestore són nombrosos[12]:

- Mentre que Firebase és simplement un JSON gegant, Cloud Firestore és forma a partir de col·leccions que contenen documents que són similars als arxius JSON i permeten una millor estructuració de dades que faciliten consultes menys profundes sense haver de consultar totes les dades. Pots encadenar filtres i combinar filtrat amb ordenament segons una propietat a la mateixa consulta.
- Pots treballar sense connexió en IOS, android i pàgines web.
- Hi ha canvis que milloren l'escriptura i les transaccions respecte Firebase.
- Té escalament automàtic que permet no haver de fragmentar les dades com s'havia de fer anteriorment.

- És més fàcil d'implementar tant si vols obtenir les dades com si vols modificar-les en temps real com si només vols obtenir la dada una vegada.
- Allotja les dades en diversos centres de dades de diferents regions, el que garanteix una escalabilitat global i una confiança sòlida.

Finalment es va decidir usar Cloud Firestore. Tot i que estava en fase Beta oferia un ventall d'opcions més ampli i més complet que l'anterior opció de Google. Alhora per la importància, les facilitats i la promoció que Google està fent, tot sembla indicar que serà la seva opció principal com a base de dades en temps real en breus. També implementa el mètode d'autentificació que ja estava present a Firebase.

## 4.3 Model de disseny: MVP + Interactor

Un cop s'ha establert la base del desenvolupament de l'aplicació era necessari determinar quina seria l'estructura de programació que s'usaria per poder reutilitzar el codi i que en cas d'implementar l'aplicació en altres plataformes mitjançant Kotlin/-Native fos relativament fàcil. S'ha decidit, entre les diferents opcions d'estructuració de l'aplicació, la que està basada en MVP i alhora implementa l'Interactor perquè s'ha cregut que s'adaptava millor al projecte.[13][14] Ha permès estructurar l'aplicació de forma que s'entén tot correctament i el codi està separat segons les seves funcionalitats. Es reflecteix un codi estructurat i que qualsevol persona, amb un mínim coneixement, pot entendre amb una curta explicació del funcionament de les classes i mètodes implementats. Es tracta d'una barreja entre diferents opcions observades a l'hora d'estructurar el programa. El resultat consisteix en:

- **View**. Té dues funcions principals: mostrar per pantalla uns elements o uns altres; i executar les funcions del Presenter associades al Model segons les interaccions que realitzi l'usuari. No implementa gairebé lògica, només la mínima necessària per decidir a vegades entre mostrar un element o un altre. Aquestes parts lògiques es troben aquí perquè si s'hagués d'usar el Presenter, aquest últim s'ompliria de codi i seria inintel·ligible.
- **Presenter**. S'encarrega de gestionar les interaccions de l'usuari i executar els mètodes del Model - Interactor que siguin necessaris. Alhora es el responsable de portar a la View les dades i demanar que es mostrin i s'executin certes funcionalitats del View. Mediador doncs entre View i Model.
- **Model**. Conté totes les classes que siguin necessàries tant per definir les funcions com les dades que s'han d'usar. Així doncs, es fa càrrec de tota la informació del programa. És doncs classe la que contindrà una part important del codi referent a l'estructura de dades i lógica interna de l'aplicació.
- **Interactor**. A part de les classes, que representen el funcionament del programa i permeten tenir la informació distribuïda com sigui necessari, tenim l'Interactor que contindrà aquells mètodes encarregats d'operar amb l'exterior del programa, ja sigui bases de dades, altres API's o el que sigui. Aquest

Interactor pot entrar en contacte directe amb el Model per desar-hi les dades que després seran la base de la lógica de l'aplicació. Es pot dir doncs que l'Interactor forma part del Model, és el seu proveïdor de dades. En l'estructura que s'ha implementat la separació entre Model i Interactor és més destacable que en altres projectes. En el present treball però, ha funcionat molt bé aquesta divisió.

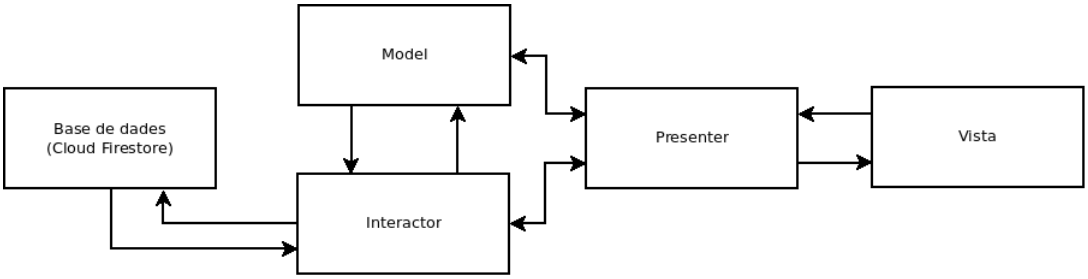


Figura 2: Flux principal de l'aplicació

En un futur, s'espera que només el *View* requereixi les majors modificacions quan es desenvolupi l'aplicació en altres plataformes. Tot i això, està clar que igualment hi haurà elements associats a certes funcionalitats que s'hauran de modificar però s'espera que siguin els mínims.

A part de tot el que ja s'ha esmentat, dintre de l'estructura s'implementarà una petita llibreria pròpia, **Utils**, on es definiran funcionalitats de caràcter genèric que serien usables en altres projectes a part del actual. D'aquesta forma es descarregarà bastant de codi el desenvolupament principal i serà més fàcil d'entendre.

## 4.4 Interfície d'usuari

En aquest punt ja s'ha parlat de les tecnologies principals que es faran servir i es passarà a explicar certes llibreries de tercers o alguns elements destacats nadius d'Android, però tot centrat en la interfície amb l'usuari(UI).

Abans d'explicar diferents tecnologies usades relacionades amb la UI és important subratllar que s'ha seguit una línia de treball pel disseny enfocada a fer-lo el més entenedor possible amb la mínima explicació. El que esta relacionat amb els motius que han donat lloc a un disseny i no un altre s'explicaran al desenvolupament.

### 4.4.1 Android: Fragments, DialogFragments i Activities

En la present aplicació s'han usat:

**Activities:** són úniques i interactuen amb l'usuari, de manera que la classe Activity s'encarrega de crear una finestra en la que es col·loquen els diferents elements de la interfície d'usuari.[15]

**Fragments:** tenen la funció de representar un comportament o una part de la interfície d'usuari en una activitat. Es poden combinar múltiples fragments en una sola activitat per crear una IU multipanell i tornar a usar un fragment en múltiples activitats. Es pot pensar en un fragment com una secció modular d'una activitat que té el seu propi cicle de vida, rep els seus propis esdeveniments d'entrada i es pot afegir o eliminar mentre l'activitat s'està executant (com una "subactivitat"que es pot tornar a usar en diferents activitats).[16]

**DialogFragments:** són un fragment que mostra una finestra de diàleg, que està sobre la finestra de la seva activitat. Aquest fragment conté un objecte Dialog, que es mostra segons l'estat del fragment. Depèn de com estigui programat es mostra aquest Dialog donant l'efecte d'un pop-up.[17]

### 4.4.2 Android: RecyclerView amb LinearLayout i GridLayout

Està clar que s'han usat Fragments, DialogFragments i Activities, però l'element propi d'Android més destacat a la present aplicació és el RecyclerView. Una vista flexible que proporciona una finestra limitada i permet mostrar un conjunt de dades gran segons s'hagin definit a l'Adapter. També s'ha d'escollir per al RecyclerView com es vol que es despleguin els diferents elements. En la present aplicació s'han usat tant LinearLayout com GridLayout per al RecyclerView:

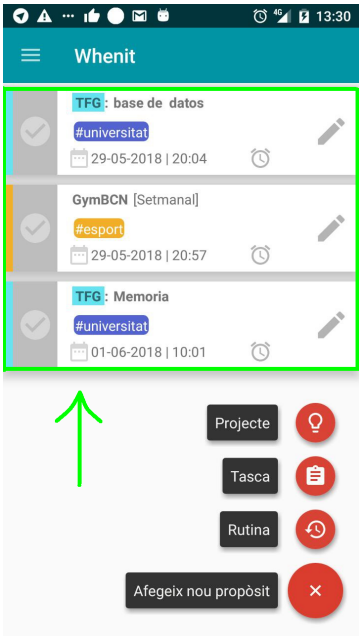


Figura 3: Vista d'un ReyclerView amb LinearLayout - d'una versió prèvia a la final


Figura 4: Vista d'un ReyclerView amb GridLayout

### 4.4.3 RecyclerView pel xat: Groupie

Per a visualitzar els missatges que s'enviïn al xat s'ha usat la llibreria Groupie[18] que permet tractar el contingut com a grups i gestiona les notificacions automàticament al RecyclerView. Facilita la gestió de les actualitzacions de contingut asíncron i de les insercions i els canvis de contingut realitzats per l'usuari. A nivell d'ítems ho simplifica tot força.

És important afegir que aquesta llibreria s'ha descobert arrel del desenvolupament del xat i tot i que no està clar si es passarà a fer servir a tots els RecyclerViews, s'ha de reconèixer que simplifica força la feina. A més, no limita l'ús del RecyclerView simplement en facilita el seu ús.

### 4.4.4 Menú: MaterialDrawer by Mikepenz

Per triar el menú desplegable es van barallar i provar diverses formes. Per una banda es va provar la versió que es genera automàticament amb Android Studio però costava canviar alguns elements després. També es va provar d'implementar la versió del menú amb els recursos que ofereix Android, però donava errors des d'un inici. Tot i que es podien haver solucionat es va trobar una llibreria que tot i que no era pròpia d'Android oferia més possibilitats i donava més facilitats a l'hora d'implementar-la. Aquesta llibreria s'anomena MaterialDrawer i ha estat desenvolupada per Mike Penz.[19] Ha permès no perdre tant de temps havent d'implementar el Drawer per defecte i ha proporcionat una opció molt més flexible que facilita modificar els elements continguts al menú.

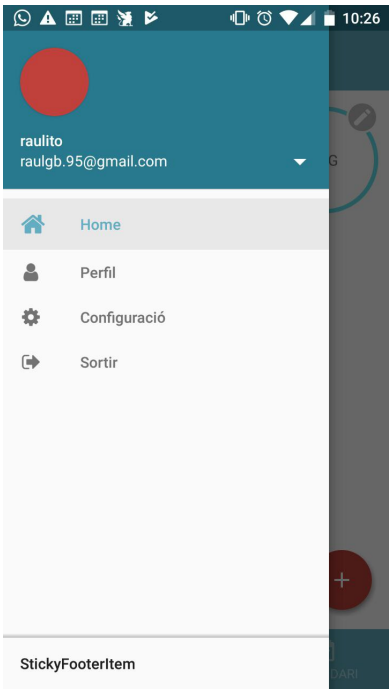


Figura 5: Vista del menú amb MaterialDrawer - d'una versió prèvia a la final

### 4.4.5 FloatingActionButton amb Clans

S'ha escollit el FloatingActionButton desenvolupat per Clans[20] perquè donava més opcions al desenvolupar aquest element al aplicatiu. Es pot veure en funcionament:

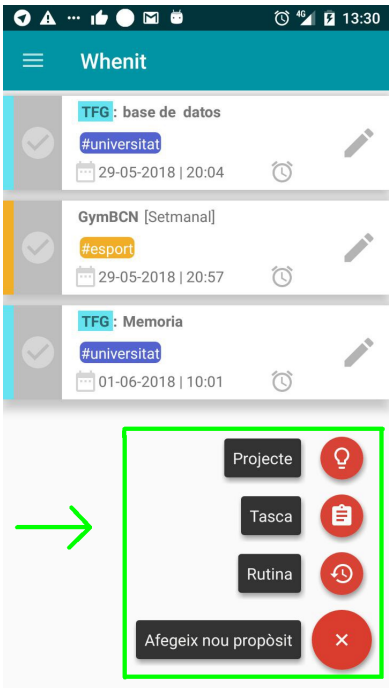


Figura 6: Vista del floating action button apretat(d'una versió prèvia a la final)

### 4.4.6 ColorPicker amb Ambilwarna

Hi ha un parell de situacions on l'aplicació havia de requerir a l'usuari que escollis un color. Hi havia la possibilitat de programar aquesta funcionalitat i poder reutilitzar-la. No obstant es va valorar que no era necessària dedicar-li un sobresforç a implementar certes eines que ja havien estat desenvolupades per altres programadors i així es podria dedicar més temps a implementar funcionalitats pròpies de l'aplicació.

Un cop decidit que s'havia de buscar una llibreria externa amb la que es pogués escollir un color fàcilment,que alhora convergís amb el disseny que s'havia estat desenvolupant fins al moment, es va trobar *Ambilwarna*.[21] Es tracta d'una petita biblioteca que permet als usuaris seleccionar un color arbitrari d'una manera molt simple i agradable mitjançant un Dialog que es veu com un pop-up:

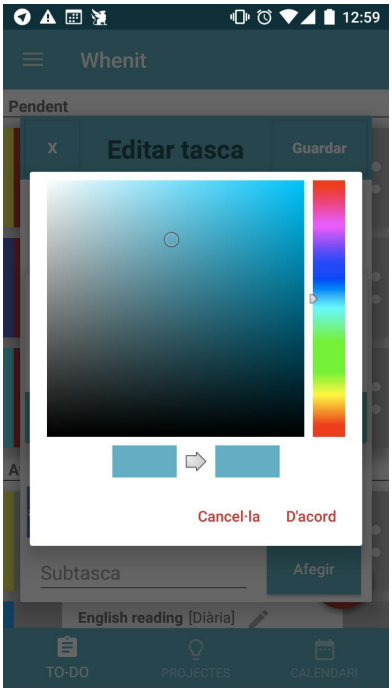


Figura 7: Vista del ColorPicker amb Ambilwarna

### 4.4.7 FlexBoxLayout

Quan a vegades s'havien d'afegir elements en algunes vistes com ara LinearLayouts sorgien problemes quan l'usuari tenia el poder d'afegir més text del comú o més categories per exemple. Està clar que es pot regular el número d'elements que pot afegir l'usuari i tractar els diferents elements amb codi propi a les diferents vistes però era un volum de codi molt costós i molt propens a errors de visualització.

A mitjans del projecte es va valorar que no sortia a compte haver de modificar aquesta gestió en moltes situacions i que si es podien evitar totes aquelles línies de codi i comprovacions associades a la vista hi hauria un estalvi relativament important de codi en certes parts del codi. En aquest moment del desenvolupament es va saber de l'existència de FlexBoxLayout que és una llibreria que ofereix les capacitats similars a mòdul de CSS Flexible Box Layout Module però per a Android.[22] No hi havia molta confiança per tots els problemes que s'havien hagut de gestionar anteriorment però provant-lo en les diferents situacions es va poder determinar que era capaç de resoldre un gran nombre de problemes associats a la flexibilitat d'algunes vistes i els elements introduïts. Alhora tot i que no s'ha usat per mostrar llistes que ocupin tot el Fragment, es poden usar RecyclerViews amb FlexBoxLayoutManager de forma que dona més rendibilitat perquè es reciclen les vistes enlloc d'inflar cada vista individual.

Es poden veure un parell d'exemples on es destaca la gestió automatitzada d'espai que realitza el FlexBoxLayout:

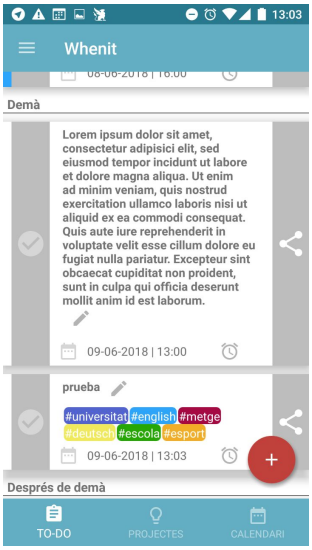


Figura 8: Vista on s'usa tant per text com per altres elements

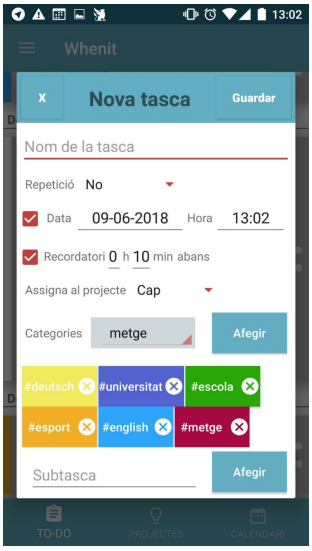


Figura 9: Vista on s'usa per botons

### 4.4.8 CircleImageView

Un cop decidit quins elements tindria la capacitat d'afegir l'usuari es va voler innovar la forma en que aquest distingiria uns elements d'altres. Tot això seguint la màxima que s'ha seguit durant tot el treball que no és altre que posar per davant la simplicitat que faciliti la comprensió abans que qualsevol altre idea a l'hora de dissenyar les vistes de l'aplicació. Per això quan va sorgir l'opció d'usar CircleImageView[23] per visualitzar els projectes es va creure que era amb diferència la millor opció al unir en un únic disseny: senzillesa, accessibilitat i estètica. Oferint una visualització més simple de tots els elements i alhora permetent que aquest ocupi menys espai i per tant que es mostrin més elements per pantalla simultàneament.

Un exemple en el projecte seria:

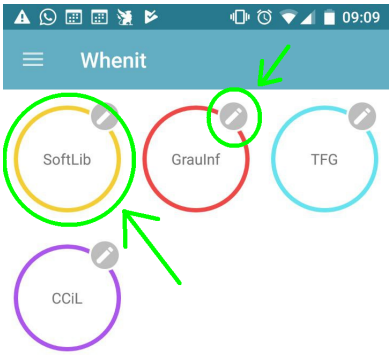


Figura 10: Vista del CircleImageView (es podria posar una imatge en el fons blanc)

# 5 Anàlisi previs al desenvolupament

Un cop definit quines tecnologíes s'han usat i abans d'explicar més profundament en quines situacions s'han fet servir s'explicarà quin va ser el raonament que va dur a definir el projecte com està i amb les aspiracions futures que ja s'han esmentat. Per això cal explicar l'anàlisi que es va fer del mercat abans d'iniciar el projecte i quins requisits es volien aconseguir amb el mateix.

## 5.1 Anàlisi de mercat

A continuació es mostraran quines han estat aquelles aplicacions en contraposició a les quals es construeix la proposta d'aplicació i disseny que fa el present treball. Tot i que s'han mira't vàries aplicacions de les quals s'ha obtingut certes referencies generals com ara Instagram, WhatsApp o Gmail, hi ha dues aplicacions que han servit de referència principal que són: per una banda el Google Calendar i per l'altre Habitica.

Ambdues aplicacions compleixen funcions diferents: una està enfocada a la calendarització, Google Calendar; i l'altre serveix per realitzar tasques i rutines d'una forma semblant a una llista. La majoria de crítiques són fruit de l'ús personal d'aquestes aplicacions durant mesos. Una altre part de les crítiques provenen de certs debats entre familiars i amics on es posaven les aplicacions per organitzar-se i sobretot les aplicacions en forma de calendaris en el punt de mira.

**Google Calendar:** és una aplicació pròpia de Google que compleix la funció principal d'un calendari. Quan s'ha d'afegir una nova tasca ofereix diferents camps: títol, data i hora, ubicació, recordatoris, convidar persones, color a escollir entre 12, afegir notes, afegir fitxers. Últimament destaca les noves possibilitats que ofereix a l'hora de marcar-se objectius i iniciar una routina. Tot i això, el que caracteritza l'app és la seva funció com a calendari. Pots compartir els esdeveniments, però no els objectius amb altres usuaris, però fins aquí es redueix la interacció entre usuaris. A part té diferents formes de mostrar el calendari, una idea força útil.

D'entre tot el que ofereix no es negarà que s'han agafat algunes idees per al present treball, però sobretot per contraposar-se, per crear quelcom alternatiu. En el calendari, la visualització principal, la mancança que s'ha trobat venia relacionada amb la incapacitat de distingir entre tasques de la mateixa família com ara podria ser les relacionades amb metges/hospitals o les relacionades amb la universitat... Després, a l'hora de crear objectius, en realitat l'únic que està fent és crear rutines, res més. El que destaca és que per a escollir objectius té un disseny molt agradable. Per tant el que es pot veure és que té un marge organitzatiu molt limitat i, tot i que mostra adequadament els esdeveniments al calendari, és incapaç de definir una forma útil i entenedora de distingir-los per a l'usuari. A part, la forma en com l'usuari visualitza les seves rutines no és molt útil i si en tingués varies se li ompliria el calendari i no seria gens còmode la pantalla principal. Per sobre de l'anterior problema, no es

poden afegir subtasques, petites tasques associades a una altre. Google Calendar només permet afegir una nota.

Un exemple de l'ús d'aquesta aplicació on s'aprecia el que s'ha esmentat anteriorment:

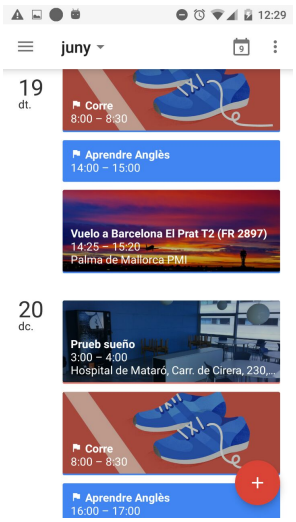


Figura 11: Usant Google Calendar [24]

Es pot veure com per una banda les tasques repetitives ocupen molt. A més, tot i que mostra imatges en el fons dels esdeveniments, i això ajuda a determinar de que es tracta, no hi ha forma de saber quina mena de tasca és fàcilment(metge, viatge...).

**Habitica:** es tracta d'una aplicació enfocada a organitzar i planificar el temps a nivell individual com si es tractés d'un joc RPG. La part relacionada amb el joc pot ser més o menys interessant però és una qüestió més subjectiva relacionada amb el gust i per això no es valorarà aquestapartat. D'altra banda, té un format organitzatiu a l'hora de visualitzar les tasques força útil on es prescindeix de qualsevol mena de calendari per prioritza la visualització d'aquelles feines pendents que s'han de fer. Alhora la forma que té per mostrar les tasques rutinàries a nivell diari permet tenir un ordre molt més clar que no pas amb Google Calendar. A més es poden afegir subtasques associades a altres tasques.

Està clar que també té mancances i, vàries, força importants. Sobretot, la impossibilitat de compartir aquestes tasques amb ningú altre, restringint el seu ús a nivell personal, l'únic que té són fòrums i una funcionalitat perquè algú pugui copiar els objectius d'un altre, però en cap cas és funcional. Alhora no té cap forma de visualitzar les tasques a nivell de calendari, el que dificulta poder valorar el volum de feina pera organitzar-se i planificar-se a mitjà i llarg termini. A part de la falta de calendari i de no poder compartir tasques hi ha un altre obstacle que minva les possibilitats d'Habitica per ser una eina útil per què els usuaris puguin balancejar el temps per conquerir els seus objectius. Aquesta dificultat és que no ofereix l'opció

d'afegir objectius a part de les tasques diàries. D'aquesta manera, la llibertat de l'usuari es veu reduïda a crear rutines únicament diàries o tasques puntuals, però no pot sortir d'aquesta dualitat. A a més de no poder definir objectius tampoc es poden associar tasques a cap categoria.

A nivell de disseny s'ha valorat força positivament ja que és força simple i permet organitzar tasques a curt termini, com es pot veure:

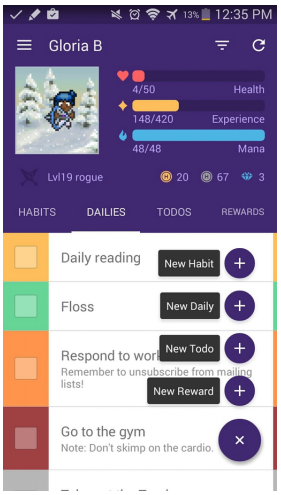


Figura 12: Habitica en funcionament [25]

## 5.2 Anàlisi de requisits

Com a conseqüència de les mancances d'altres aplicacions s'ha pogut complementar el funcionament de l'aplicació afegint objectius i redefinint certes idees de disseny que al usar aquestes aplicacions s'han hagut de canviar. Finalment els requisits que han acabat marcant les línies de treball han estat els següents:

**Autentificació:** l'usuari es podrà registrar de dues maneres, la primera i principal serà a partir del seu compte de Google i l'altre serà mitjançant el correu electrònic. L'usuari en cas d'haver-se registrat a partir d'un correu electrònic haurà de poder recuperar la contrasenya. A més a més, l'usuari haurà de poder recuperar les seves dades en cas d'obrir el seu compte amb un altre mòbil a partir de la base de dades creada amb Cloud Firestore. També permetrà que si l'usuari inicialment s'ha registrat manualment però després utilitza el mateix correu per entrar amb l'opció de Google pugui entrar al seu compte anterior.

**Categories:** aquesta funcionalitat que implementarà el projecte és única, o com a mínim, no s'ha vist a cap altre aplicació coneguda. Les categories permetran establir a l'usuari una manera d'identificar certes tasques amb una etiqueta com podria ser *Universitat*, *Metge* o qualsevol altre que l'usuari vulgui definir. D'aquesta forma es

veurà clarament les tasques que són de la mateixa família i quedarà un disseny amb colors que ho facilitarà.

**Tasques:** s'haurà de poder afegir tasques i que el creador pugui establir: el nom; si vol que es converteixi en rutinari o no; definir una data i una hora; escollir i definir un recordatori; associar-ho a un projecte entre els que ha pogut arribar a crear; afegir categories; i afegir subtasques. Tot això s'ha de poder definir d'una forma fàcil i entenedora i que l'usuari pugui entendre-ho i visualitzar tot sense que se li solapin la visió d'uns elements obre altres.

**Rutines:** tenen les mateixes opcions que les tasques però l'usuari està obligat a definir una rutina que podrà ser: diària, que es faci cada dia; setmanal, que es realitzi els dies de la setmana que l'usuari vulgui; mensual, on l'usuari podrà decidir el dia del mes que vol realitzar la tasca; i anual, on escollirà quin dia de l'any es repeteixen les tasques.

**Projectes:** aquests seran el nucli del projecte entorn al que gira la nostra aplicació. Es tracta d'un concepte innovador que pretén oferir la possibilitat d'iniciar projectes vitals o de feina al voltant dels que associar tasques o rutines i així poder organitzar-se. És innovador en tant que l'usuari és completament lliure a l'hora de definir els projectes i repartir-se el treball.

**Compartir:** aquest apartat serà el més bàsic, serà el pròleg al desenvolupament posterior enfocat a compartir i cooperar per construir projectes col·lectivament. Inicialment s'implementarà l'opció de compartir els projectes amb altres usuaris i si dona temps un petit xat.

**Calendari:** hi haurà una vista principal que permetrà a l'usuari veure les seves tasques en forma de calendari. Aquesta visualització pretén ajudar a l'usuari a planificar-se a mitjà i llarg termini. Inicialment les opcions que s'implementaran seran la diària i la mensual. Tot i això hi ha pensada una nova forma de visualitzar les tasques a nivell setmanal que es desenvoluparà més endavant.

**Dissenys:** és el requisit que es preveu que ocupi més temps alhora que és més difícil de valorar, ja que no es pot apreciar molt bé el treball realitzat en aquest sentit. Tot i això, es creu que es l'element principal d'una aplicació com la que es proposa. Per això, per una banda es seguiran les guies de disseny marcades per material.io[26] i en cas que en algun moment s'hagi d'anar per una camí no recomanat es buscarà l'alternativa més propera a les línies marcades. D'altra banda, com es tracta d'una aplicació que es busca que sigui d'ús quotidià hi haurà una màxima en el disseny que serà la utilitat. Es vol arribar a simplificar i facilitar el seu ús per què sigui el més intuïtiu possible.

**Llengües:** inicialment l'aplicació permetrà als usuaris que usin el mòbil en català, castellà o anglès fer servir l'aplicació amb la seva llengua nativa. Es preveu que en un futur ofereixi altres llengües, però inicialment es traduiran tots els elements a les tres llengües citades anteriorment.

# 6 Desenvolupament

Un cop explicada l'estructura que definirà el marc sota el que s'ha treballat es passarà a explicar quin procés de desenvolupament s'ha seguit per poder implementar el conjunt de funcionalitats basades en en els casos d'ús dels usuaris quan encara no s'han identificat de l'apèndix A i en els casos d'ús del usuaris identificats de l'apèndix B. A més també s'explicaran concretament com s'han programat l'estructura MVP+Interactor o altres característiques importants. Alguns dels apartats contenen una explicació del disseny, la interfície d'usuari, basada en els casos d'ús i els dissenys previs en paper com els que s'aprecien a l'apèndix F. S'ha obtingut una app fàcil d'usar per l'usuari i amb la que amb pocs passos pots fer una cosa o un altre, e.g. quan s'afegeixen tasques com es veu al diagrama de flux de l'apèndix C.

## 6.1 Fragments i Activities

En aquest projecte s'han fet servir Fragments per a la gran majoria de situacions i s'ha reservat l'Activity principal per a gestionar aquests Fragments. Tot i això, a l'hora d'autentificar-se s'han usat Activities i s'han gestionat les transicions entre unes o altres.

Lligat amb el que s'ha explicat en primer lloc, destaca la classe FragmentController que s'encarrega de gestionar les transicions entre Fragments i s'ha dissenyat de tal manera que en cas que s'afegeixin nous Fragments per noves funcionalitats la modificació de codi és mínima. Només s'han d'afegir on toqui els noms dels nous fragments i definir la transició. Després a l'hora d'usar-se simplement el FragmentController com a part del Presenter gestionarà la transició seleccionada per l'usuari mitjançant un botó de la View.

S'ha escollit minimitzar l'ús d'Activities i usar Fragments perquè són fàcils les transicions i menys costoses. D'aquesta manera es simplifica la interacció entre Fragments mitjançant la MainActivity des d'on es cridaran els Fragments.

## 6.2 MVP + Interactor

En el projecte s'ha desenvolupat una estructura de carpetas i de classes que funciona segons el plantejament MVP que s'ha explicat anteriorment, alhora s'ha implementat l'Interactor dintre del Model. Tot això ha estat gràcies també a la interfície definida a MainMVP, que defineix les relacions entre Model, Vista i Presenter, i també les funcions comuns a totes les classes corresponents. Fins al moment no s'han hagut d'implementar funcions comuns. Sobre el funcionament que segueix l'aplicació es destacaran algunes parts positives i altres que han resultat problemàtiques. En tot cas, després d'haver desenvolupat l'aplicació segons els objectius és molt remarcable les facilitats que ha donat aquesta estructura a l'hora de programar l'app.

S'ha programat com s'ha explicat anteriorment però a nivell del paquet **View** dintre d'algunes classes com la MainActivity ha estat necessari implementar un seguit de mètodes que es criden des dels Fragments per tal d'accedir a les funcionalitats que ofereix el Presenter i el Model. Dintre del View es col·locaran els adapters que serveixen per als RecyclerViews. Així doncs ha quedat sobrecarregat, no tant de codi perquè en si no hi ha molt de codi dintre dels mètodes, sinó perquè té forces funcions que no són pròpies de la MainActivity però que si que és on han d'estar perquè són usades pels Fragments entre els que alterna la vista de la MainActivity. Aquest ha estat el problema principal de la vista. A part, tot i que no es volien usar Ifs a cap de les classes del paquet View no ha estat possible. Hi ha hagut situacions, com algunes associades als Adapters que s'usen pels RecyclerViews, on es requeria comprovar algunes variables dels elements que després es mostrarien per pantalla i per tant era obligatori fer les comprovacions pertinents. Tot i que hi ha algunes excepcions, s'ha filat força prim a l'hora de seguir l'estructura MVP+Interactor.

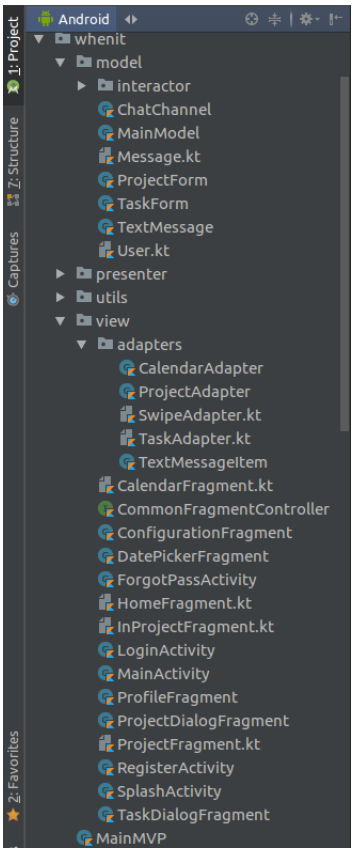


Figura 13: MVP + Interactor

Com a conseqüència del problema de la vista el **Presenter** està força buit. Tot i això, implementa algunes funcionalitats i serveix de mitjancer entre les classes del View i el Model. D'aquesta manera es pot afirmar que tot i que no s'estigui explotant l'ús del Presenter com tocaria es manté el seu ús i permet orientar-se dintre de l'estructura de l'aplicació.

També hi ha una classe Utils.kt que esta dintre del paquet *utils* que implementa mètodes genèrics que es reutilitzen a diferents parts del codi bastant dispars treient càrrega de codi d'algunes classes i mètodes.

Finalment, és important destacar que ha estat de molt útil la forma en com s'ha estructurat el **Model** i l'**Interactor**. La divisió ha servit per no sobrecarregar el Model i també per separar clarament les interaccions amb la base de dades de les dades en ús del model, i així s'ha facilitat molt la implementació de tots els mètodes relacionats amb les dades.

## 6.3 Autentificació

S'ha usat la tecnologia que ofereix Cloud Firestore per a l'autentificació dels usuaris. Hi ha mètodes propis de Firebase que permeten el registre d'usuaris i la identificació dels mateixos deixant pas a la pantalla principal o no.

Hi ha dues maneres d'accedir a l'aplicació: a partir dels comptes de Google que es tinguin registrats al mòbil Android o a partir d'un correu electrònic que l'usuari decideixi. Per fer-ho mitjançant el correu electrònic t'has d'haver registrat prèviament fent servir l'opció que s'ofereix. En cas d'oblidar-te la contrasenya es podrà recuperar. Hi haurà pantalles, Activities, tant pel registre com per la recuperació de la contrasenya i s'usaran funcionalitats pròpies de la base de dades per gestionar les dues opcions. Es comprovarà, a l'hora de registrar-se, que l'usuari hagi introduït les dades correctament i com a mínim una contrasenya major que 8 i que contingui números i lletres. Quedarà doncs de la següent manera:

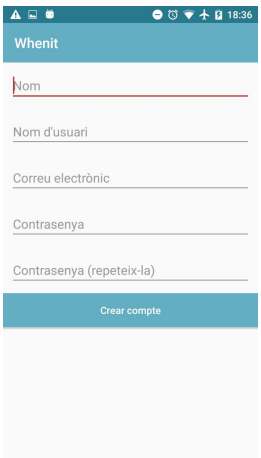


Figura 14: Vista on es registra l'usuari

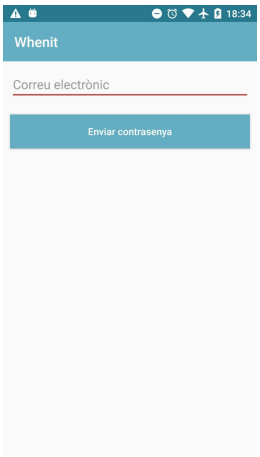


Figura 15: Vista on es recupera la constrasenya

D'altra banda tenim també l'Activity per loguejar-se inicialment on l'usuari pot accedir al seu compte directament mitjançant el seu compte de Google o en cas de que ja s'hagi registrat pot accedir-hi introduint les dades corresponents:

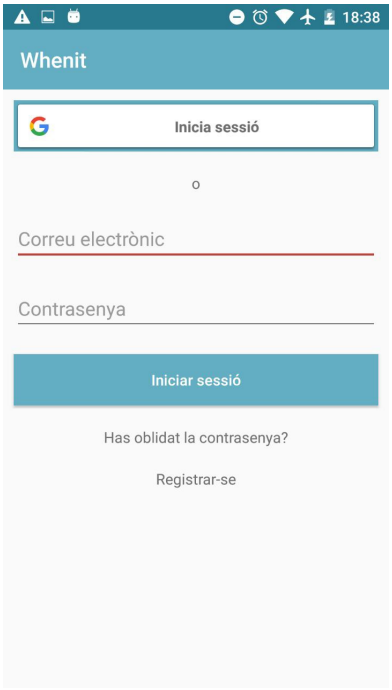


Figura 16: Vista inicial per accedir a l'app

També hi haurà una SplashActivity que es mostrarà mentre l'aplicació comprova si l'usuari ja està registrat i ha iniciat sessió anteriorment, sense haver-la tancat, amb aquest dispositiu. En cas negatiu entrarà a l'inici, en cas que ja hagi iniciat sessió anteriorment i estigui encara iniciada accedirà directament a l'aplicació. Donant fluïdesa a l'ús de la mateixa.


Figura 17: Vista de la SplashActivity

## 6.4 Menú amb MaterialDrawer

A continuació d'haver implementat les diferents Activities s'ha usat la llibreria MaterialDrawer a la MainActivity. Permet a l'usuari accedir al menú fins i tot si no es mostra la barra superior, gràcies a la lliscament d'esquerra a dreta des del cantó esquerra. En aquest punt, per exemple, s'ha usat el FragmentController per canviar del Fragment de la pantalla principal als que ofereix el menú. Es tracta d'un menú embrionari però amb el disseny ja funcional:

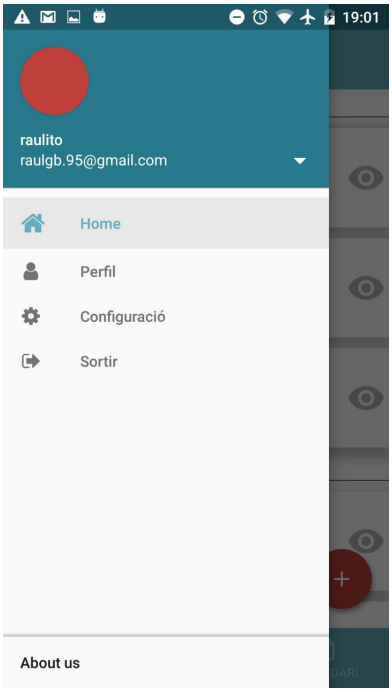


Figura 18: Vista del menú

## 6.5 Afegir Tasques, Rutines i Projectes

La funcionalitat bàsica és afegir plans i s'ha buscat marcar la diferència respecte altres aplicacions en la forma en com s'afegeixen les tasques, rutines o projectes, perquè es fa mitjançant un DialogFragment. En tots els Fragments, menys en els propis del menú o dintre els projectes, l'usuari tindrà l'opció d'afegir qualsevol dels elements que s'han referit anteriorment a partir del FloatingButtonAction que ja s'ha explicat com es mostra a la figura 6.

L'usuari intentarà afegir qualsevol dels elements i se li apareixerà un DialogFragment que en realitat és diferent entre Tasques i Projectes però no entre Tasques i Rutines. A més, una tasca pot estar redefinida per a que sigui una rutina. Les opcions que té l'usuari a l'hora d'afegir una tasca donen un marc de llibertat superior respecte altres aplicacions. Alhora les rutines tenen unes opcions i forma de presentar-se que simplifiquen la forma en com s'introdueixen les repeticions de forma molt intuïtiva.

A nivell de Tasques i Rutines s'ofereixen les opcions definides als requisits com es pot veure:

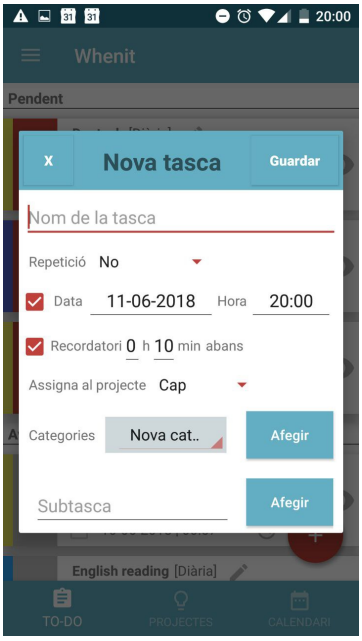


Figura 19: Dialog per afegir tasca, sense repetició

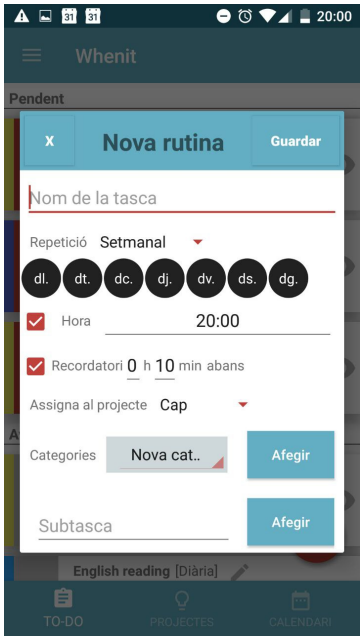


Figura 20: Dialog per afegir rutina, seleccionada l'opció setmanal

A part existeix un altre DialogFragment que permet afegir un Projecte que després es pot assignar a qualsevol tasca o rutina. Alhora es pot compartir amb altres usuaris:

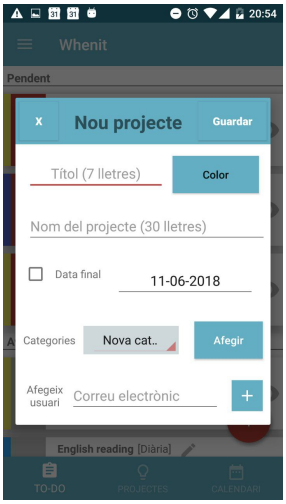


Figura 21: Dialog per afegir Projecte

A part hi ha els DialogFragments que s'obren al editar qualsevol de les tasques, rutines o projectes; però són gairebé idèntics.

Quan l'usuari guarda qualsevol dels elements que està creant o editant es produeix una interacció amb la base de dades, però gràcies a l'asincronia d'aquestes interaccions no es bloqueja el fil principal. En el moment en que comença a crear una tasca, les rutines usen la mateixa classe que les tasques, o un projecte es creen objectes de les classes TaskForm o ProjecteForm corresponents. Si al final es guarda es realitza un parseig entre aquesta classe i el Document que es guarda Cloud Firestore, que seria una espècie de JSON. En canvi si es cancela la creació simplement s'elimina.

## 6.6 Editar les tasques, rutines o projectes

Un cop l'usuari es planifica, cap la possibilitat que desitgi modificar qualsevol de les tasques, rutines o projectes, plans. Per això, com ja s'haurà vist en algunes imatges, hi ha una figura d'una llapis/boli que l'usuari identifica fàcilment amb l'edició. D'aquesta manera quan l'usuari premi s'obrirà un diàleg semblant al de creació i l'usuari podrà editar com ell desitgi aquestes tasques:

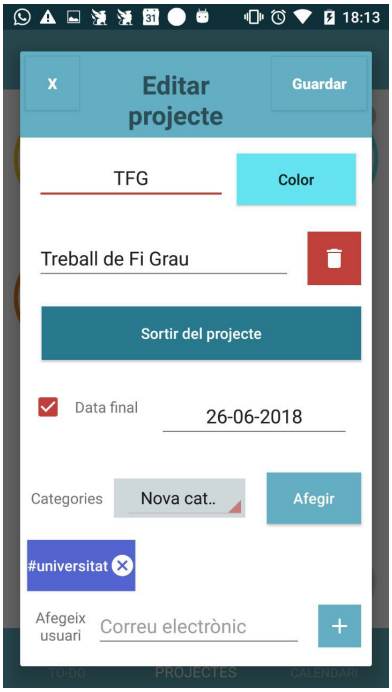


Figura 22: Vista d'una edició d'un projecte

Dintre de les opcions que ofereix l'edició es diferència de la creació la possibilitat d'eliminar qualsevol mena de plans, i en cas d'estar en un projecte alhora es permet sortir del projecte sense eliminar-lo. Quan s'elimina s'esborra de la base de dades, abans de fer-ho li pregunta a l'usuari si ho vol fer. En canvi en el cas de sortir del projecte només elimina la seva vinculació amb el projecte.

Mentre està editant l'element TaskForm, tant per tasques com per rutines, o el ProjectForm, per projectes, té una còpia dels que està editant i només quan ho guardà es realitza per una banda la modificació de la base de dades i per l'altre del model local. Més endavant s'explicarà com s'obtenen les dades pel model local.

És destacable que a l'hora d'editar una rutina es pot editar la tasca concreta d'un dia o la rutina completa. Això és possible gràcies a un Dialog, intermedi, que obliga a escollir quina de les dues opcions es vol modificar:

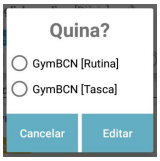


Figura 23: Dialog per escollir

## 6.7 Rutines

Aquestes requereixen una explicació pròpia perquè han afegit varies dificultats que a l'aplicació. Al generar una rutina l'usuari només estaria generant una tasca però a l'hora de visualitzar aquesta rutina al seu dia a dia serien moltes tasques "idèntiques". El que s'ha fet per solucionar-ho és generar una tasca temporal que l'usuari visualitza quan ho necessita però que no s'afegeix a la base de dades a no ser que l'usuari l'editi o que la marqui com a realitzada. S'afegeix a la llista de tasques que s'envia al RecyclerView amb un identificador format a partir de la data que li corresponia i l'identificador de la rutina, però encara no s'afegeix a la base de dades.

Per tractar tot això, quan s'obté de la base de dades la llista de tasques i rutines, es defineix la següent rutina, que serà la que es mostrarà a la vista inicial, TO-DO. En cas que aquesta ja existeixi, és a dir, que ja hi hagi una tasca que prové de la base de dades que correspongui a la nova tasca temporal que la rutina proposa crear no s'afegeix aquesta tasca temporal perquè ja està la de la base de dades. Després, al calendari, es mostren tant les tasques que no s'han realitzat com les que si a la vista dels dies corresponents. Si s'elimina la rutina no s'esborren les tasques que s'han guardat o editat que pertanyien a la rutina. Les tasques que es guarden a la base dades que provenen de rutines no es poden convertir en rutines per tant no són tasques genèriques i tampoc són rutines perquè es tracta d'una tasca concreta, així doncs estan entre tasca i rutina i alhora es permet la seva edició.

## 6.8 Categories

Cada usuari té associades unes categories entre les que pot classificar les tasques o els projectes. D'aquesta manera hi ha una visió agradable de les tasques que estan

relacionades. Si l'usuari vol, pot crear una nova categoria al mateix DialogFragment on estigui definint quines categories té la tasca, rutina o projecte. Aquestes categories es guarden a la base de dades i gràcies a això es manté el color en tot moment.

## 6.9 Barra inferior

Per a la barra inferior s'ha usat BottomNavigationView[27] que presenta una barra de navegació inferior estàndard per a l'aplicació. Permet que els usuaris puguin canviar entre vistes amb un únic click. Encara que no es definitiu, el plantejament és que segueixi el model definit per material.io i no hi hagi mai més de 5 elements.

## 6.10 TO-DO - Vista inicial

La vista principal on l'usuari aterrà quan inicia l'aplicació mostra les tasques i rutines que l'usuari té pendents, tant les que ja tenen la data vençuda com les que tenen dates futures o s'han de realitzar el mateix dia. Per a fer això s'utilitza un RecyclerView amb LinearLayout, però abans d'introduir la llista de tasques pendents s'han de tractar varis obstacles per aconseguir una vista no sobrecarregada com:

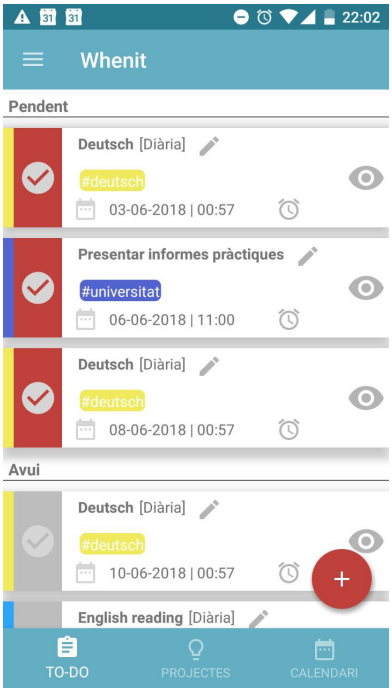


Figura 24: Vista inicial - TO-DO

Es mostren les tasques dividides per quatre separadors. Per aconseguir això s'ordenen les tasques segons els dies i s'afegeixen algunes tasques que simplement són els separadors. Després l'Adapter corresponent s'encarrega de gestionar aquestes tasques per mostrar els elements com sigui pertinent. El primer separador és el de

les tasques pendents anteriors a avui, que s'indiquen com a pendents i es visualitza de color vermell el botó per indicar que està feta. El segon és el d'*avui*, on es mostren les tasques ja sense el color vermell. Després es mostra el *demà* i l'últim separador després del que es col·loquen totes les tasques restants és el de *després de demà*.

D'aquest **disseny** destaquen tres elements. En primer lloc es mostren les tasques que encara no estan fetes, i d'aquesta manera l'usuari no veu sobrecarregada la vista inicial i només hi veu la feina que li queda per fer. En segon lloc, només es mostren la tasca/esdeveniment relacionat amb el següent dia que cal realitzar la rutina, en cap cas es mostren totes les rutines futures en la vista inicial, descarregant així aquesta vista. En darrer lloc es vol destacar que en aquesta mateixa vista l'usuari pot marcar com a realitzades les subtasques ja que al prèmer sobre una tasca si hi ha subtasques es despleguen les mateixes i apareix un "Checkbox" associat a cadascuna que pot ser marcat.

En un futur s'implementarà algun altre mena de vista inicial a part de l'actual que faciliti veure més número de tasques. Tot i això es mantindrà com a opció perquè és una forma agradable de visualitzar la feina.

## 6.11 Projectes

Els projectes com s'ha explicat anteriorment es guarden a la base de dades realitzant un parseig de la classe ProjectForm. Tot i això, es guarda un altre element, *projectUser*, que relaciona projectes amb usuaris. Després, si es guarda una tasca relacionada amb un projecte es guardarà un element a *projectTask* que la relacioni amb un projecte.

Tot i que no semblava possible finalment s'ha aconseguit implementar una forma cooperativa de crear projectes. L'app funciona per a grups/projectes de forma satisfactòria. D'una banda, es poden afegir usuaris als projectes ja creats, a partir del correu usat per autenticar-se, dintre del Dialog de l'edició dels projectes. D'altra banda, es poden visualitzar les tasques associades a un projecte i existeix un *xat*, explicat en *6.15 Xat*, del que poden participar els diferents integrants del projecte. Per fer tot això, ha calgut un disseny previ de la base de dades conscient dels objectius futurs i també un ús adequat del sistema que ens oferia Cloud Firestore com a base de dades, s'explicarà en el punt *6.14 Base de dades: Cloud Firestore*. Però resumint-ho breument es pot dir que s'ha aconseguit una correspondencia entre el model i la base de dades i s'ha fet un bon ús de l'estructura de classes MVP+Interactor. Obtenint com a conseqüència un codi que si es mirés la progressió de dedicació temporal en relació a les funcionalitats implementades es podria apreciar una corba decreixent. Així doncs, es podria dir que al final del projecte s'han pogut implementar, mirant tutorials i adaptant-los al projecte, funcionalitats com el xat o la visualització mensual amb un temps molt menor del que s'hagués requerit si es tractés d'un codi sense estructurar.

Prement a sobre del CircleView d'un projecte s'accedeix a una nova vista que permet veure totes les tasques associades a un projecte:

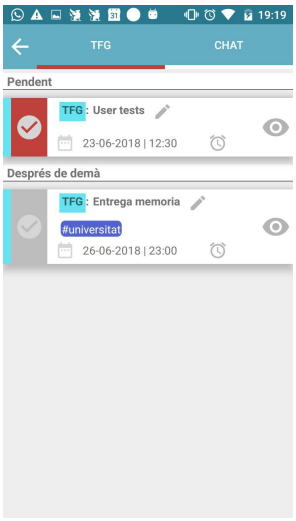


Figura 25: Vista de les tasques del projecte TFG

I alhora també permet entrar al xat i enviar-hi missatges(s'espera que en un futur es puguin enviar altres elements a part de missatges):

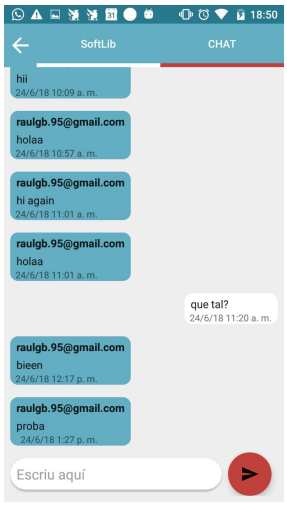


Figura 26: Vista del xat del projecte SoftLib per raulgb.pokemon@gmail.com

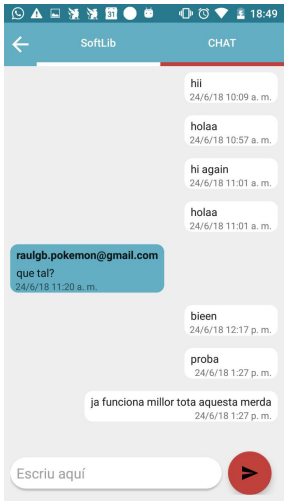


Figura 27: Vista del xat del projecte SoftLib per raulgb.95@gmail.com

D'aquesta manera els usuaris que han estat afegits a un projecte visualitzen i afegeixen tasques igual que els que han creat el projecte mentre poden participar del xat i comunicar-se entre ells. Tot i que encara no hi ha una gestió dels permisos, en un futur serà necessària per a que pugui existir un o més d'un administrador dels projectes que pugui regular com vulgui el projecte. La base de dades queda de la següent manera:

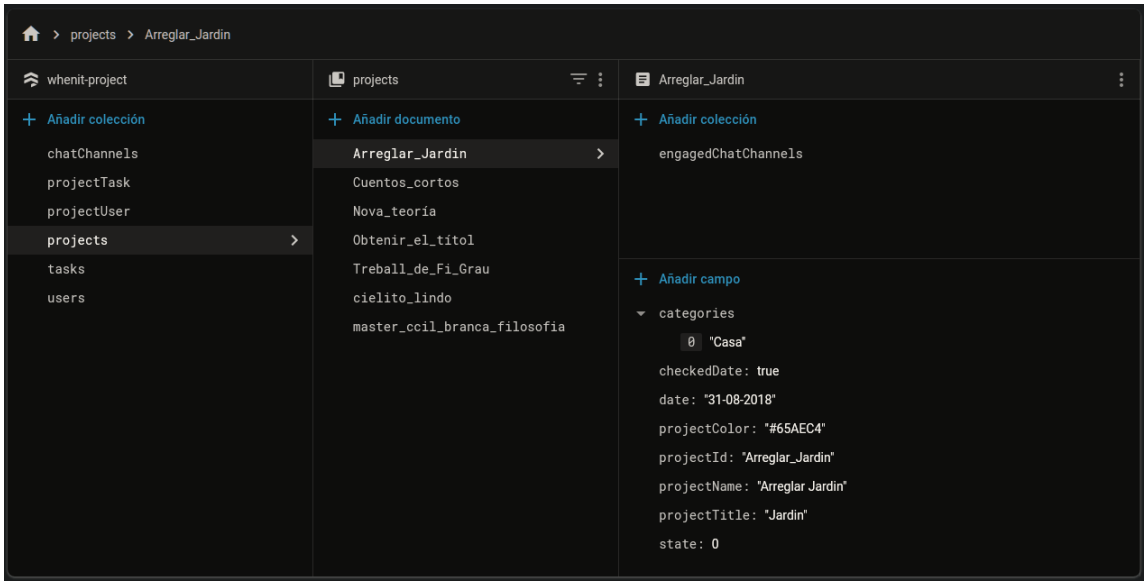


Figura 28: Base de dades

D'altra banda, quan una tasca està relacionada amb un projecte queda d'aquesta manera:

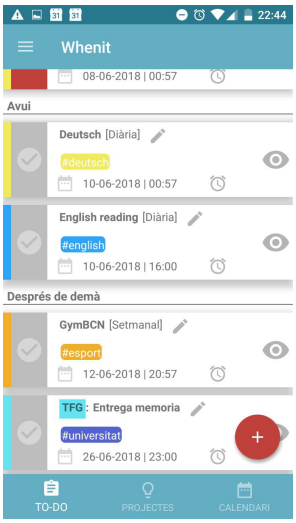


Figura 29: Tasca associada al projecte TFG

## 6.12 Vista dels projectes

Aquesta vista permet visualitzar els diferents projectes dels que un usuari participa, editar-los i entrar-hi. S'obre el xat al prémer al CircleImageView relacionat amb el projecte pertinent. En un futur es podrà afegir una imatge al CircleImageView associat amb un projecte i també es veurà amb un numerò en petit el número de missatges pendents. A continuació es veuen les imatges de la vista dels projectes:

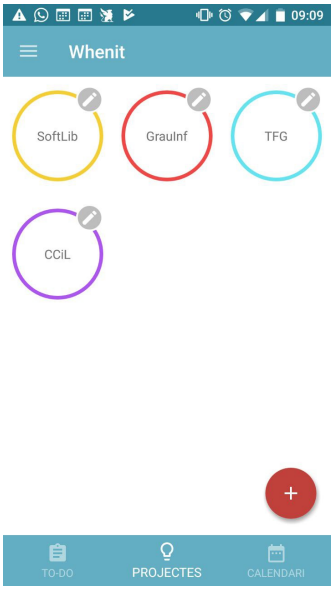


Figura 30: Vista dels projectes

Sobre el **disseny** es destaca l'ús que ja s'ha esmentat del CircleImageView però també destaca que en aquest RecyclerView en comptes d'un LinearLayout s'ha usat un GridLayout i així l'usuari pot veure més projectes a la mateixa vista simultàniament.

## 6.13 Vista del calendari

Aquesta és la última vista principal i ha requerit el desenvolupament d'una capacitat pròpia de gestionar el temps per a poder mostrar en un calendari les tasques i rutines que l'usuari tingui guardades.

En termes generals funciona igual que la vista TO-DO però en aquest cas només es visualitzen les tasques relacionades amb el dia o dies que es mostren al calendari en cas de que sigui un dia o un mes respectivament. Per fer això abans s'han de tractar la llista de tasques i rutines que l'usuari té a nivell local i mostrar les que corresponen als dies que l'usuari vol visualitzar. El major problema es troba a l'hora de visualitzar les rutines, ja que s'ha de tractar el temps de forma que es comprovi per cada rutina si els dies que es volen mostrar corresponen als dies que toca realitzar la rutina, en cas afirmatiu o si anteriorment no hi havia cap tasca de la rutina associada a aquesta data a la base de dades es genera una tasca temporal relacionada amb aquest dia i la rutina pertinent. Tot això es realitza cada vegada que es visualitzen uns dies o altres, però la forma en com es tracta i al tenir les dades en local un cop ja iniciada l'aplicació no té masses costos computacionals.

Per aconseguir aquest efecte ha estat important definir una forma de carregar les dades en el Fragment d'aquesta vista per què en cada cas es visualitzin les tasques d'una manera o d'una altre. És a dir, depenent de si l'usuari prefereix veure l'opció

diària o mensual es veuran diferent.

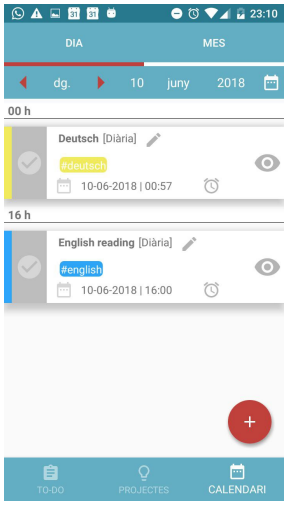


Figura 31: Vista del calendari Dia


Figura 32: Vista del calendari Mes

El **disseny** elimina la barra superior i permet escollir entre dos modes de visualització del calendari sense sobrecarregar aquesta vista. Si es visualitza al disseny mensual es poden veure les tasques pendents i les que ja s'han fet associades als dies que corresponen. Alhora, s'aprecien els dies relacionats amb el mes que s'està mirant i el dia en el que es troba es mostra d'un color diferent. En canvi els dies d'altres mesos es mostren amb uns fons verdós tot indicant que aquests dies ja no pertanyen al mes que s'està visualitzant. L'usuari al prémer en un dels dies anirà directament a la visualització diària. En un futur s'implementarà un mode setmanal que oferirà una nova forma de veure les tasques.

## 6.14 Base de dades: Cloud Firestore

Per usar firestore primer s'ha hagut de vincular el projecte de Cloud Firestore a Android Studio tot seguint el camí que ofereix la pròpia eina de *Google*. Un cop vinculat amb el projecte s'han d'afegir les llibreries associades amb Firestore a les dependències per tal de podar usar els mètodes que ofereixen. Arribats a aquest punt ja es pot requerir l'autentificació, fer crides a la base de dades o *queries* fent servir els mètodes que proporcionen aquestes llibreries.

El conjunt de mètodes que interaccionen amb la base de dades estan inclosos al paquet **Interactor** i principalment el trajecte que realitza la informació és el que s'ha mostrat a la figura 2.

A continuació s'aprofundirà sobre com s'ha estructurat la base de dades i perquè i què s'implementat concretament al Interactor.

### 6.14.1 Interactor

Per facilitar la interacció amb la base de dades ha anat molt bé definir tots els tipus d'interaccions amb aquesta en diferents mètodes en un parell de classes, *SimpleInteractor* i *MainInteractor*, contingudes a la carpeta Interactor. La primera classe només s'encarrega de les funcions relacionades amb l'autentificació inicial. En canvi la segona s'encarrega de totes les altres funcions que van des de l'obtenció inicial de les tasques, rutines i projectes que l'usuari ha format o dels que forma part fins a l'obtenció i l'enviament de missatges en el xat d'un projecte concret. Es podria subdividir en diferents classes aquesta última, però tampoc s'ha cregut necessari crear moltes classes i després torbar-te amb la necessitat d'anar creant objectes tota l'estona.

La majoria de vegades que l'usuari accedeix a la base de dades o que es modifica la base de dades i el Listener de l'aplicació ho detecta les dades es guarden al Model. Aquest model conté llistes de tasques, rutines i projectes i està en local. D'aquesta manera es facilita molt el treball amb aquestes dades en els casos on s'han de mostrar a l'usuari. Ara per ara, no s'ha cregut convenient guardar totes les dades dels xats al Model però no hi hauria problema en fer-ho.

### 6.14.2 Base de dades

A l'hora d'explicar la base de dades hi ha dues parts importants:

- **Programació de l'interacció:** Per fer servir la base de dades hi ha un obstacle que s'ha hagut de sobrepassar, l'asincronia. Per treballar amb aquesta s'ha fugit del sincronisme i de les esperes i s'ha hagut de replantejar el funcionament del programa en alguns punts per poder desenvolupar el codi correctament. Ha estat cabdal adonar-se que després de realitzar les *queries* a la base de dades el programa continua funcionant i l'usuari pot prémer qualsevol botó. Tot i això, hi ha moments que es necessiten les dades, el que s'ha fet és carregar la pantalla buida i l'usuari ha d'esperar a que es carregui la informació. Tot i això, l'usuari pot canviar de pantalla mentre es carreguen. Mentre l'usuari està interactuant amb l'aplicació arriben les dades i es carreguen. Per a poder carregar-les a la pantalla, és necessari haver carregat algun mètode dintre la *query* per què quan arribin les dades s'executi i es mostri on i com toqui la informació a l'usuari.

  Més concretament hi ha varies funcions de la llibreria de Firestore que s'han usat de diferents maneres. Per una banda, les relacionades amb l'autentificació i la creació d'usuaris que es troben a la classe *SimpleInteractor*. Per l'altre, a la classe *MainInteractor* es troba un ampli ventall de funcions: les que creen tasques, projectes o rutines segons l'existència d'aquests a la base de dades; altres que obtenen les tasques, rutines i projectes de la base de dades i ho guarden al model o executen funcions que els hi són donades per el mètode que les crida; unes que específicament esborren tasques, rutines o projectes

respectivament; i finalment hi ha les funcions que s'encarreguen de gestionar la creació del canal per als missatges als projectes i altres que gestionen l'enviament i la obtenció d'aquests missatges.

Alhora, per obtenir les funcions es pot fer tant obtenint l'objecte amb la funció *get* o mitjançant *addSnapshotListener* per obtenir la informació alhora que s'afegeix un *listener* que permetrà saber quan es modifiquen les dades de la base de dades i actualitzar la vista. A part, per rebre els projectes o altres elements de forma eficient de la base de dades s'usen *queris* que permeten aconseguir els **documents**, forma en la que es guarden els objectes a la base de dades, mitjançant el mètode *whereEqualTo* ,que retorna només els documents si el valor dintre del camp que s'especifiqui és el mateix que el que se li entrega, o altres.

- **Estructura de la base de dades:** Per tal de poder accedir de la millor forma a la base de dades, és a dir, tant a nivell d'eficiència com per a que sigui fàcil d'entendre s'ha definit una estructura concreta que consta de diferents col·leccions i on els documents tenen uns camps concrets. Primer hi ha les **col·leccions**: *chatChannels*, *projectTask*, *projectUser*, *projects*, *tasks*, *users*; on es defineixen on es dipositaran els diferents conjunts de *documents* amb els respectius camps a cadascun. Alguns *documents* com els que estan relacionats amb els projectes contenen **subcol·leccions** que poden contenir altres *documents*, en aquest cas aquesta *subcol·lecció* està relacionada amb el xat. Cada projecte té a la base de dades una *subcol·lecció* que està directament relacionada amb un xat que està a la col·lecció *chatChannels*. D'aquesta manera al tenir l'identificador del projecte es pot accedir directament al document del xat. Els projectes podrien tenir un identificador aleatori, ara per ara s'assigna respecte al nom del projecte. Tot això es pot veure a la figura 33.

Hi ha una funcionalitat que es gestiona automàticament, gràcies a Cloud Firestore, però que s'ha cregut important destacar i és la persistència de les dades quan no hi ha connexió i tot el que això comporta. Per una banda permet treballar sense connexió i que l'usuari mentre no tingui Internet pugui estar visualitzant les tasques i totes les dades **sense connexió**. Per l'altre l'usuari té la possibilitat de penjar dades, ja sigui creant nous plans o enviant missatges, que es penjaran quan l'usuari un altre cop tingui connexió, actualitzant-se amb les dades que altres usuaris hagin pogut modificar.

### 6.14.3 Costos

Al usar aquesta eina que proporciona Google s'està avocat a haver d'assumir les seves condicions sobre l'ús i els costos d'utilitzar aquesta eina. D'altra banda, ofereix una robustesa i un conjunt de possibilitats molt més grans de les que mai es podrien arribar a implementar amb una base de dades pròpia. Fins ara s'ha valorat que aquesta és la millor opció, caldria però revisar quines són les seves condicions i

fins a quin punt és beneficiós pel projecte. També s'hauria de valorar com es poden contrarestar les seves exigències a nivell de pagaments. Tot sembla apuntar que en un futur s'haurà de tenir un pla d'ingressos per poder fer front tant a despeses de Cloud Firestore com d'un altre caràcter. Les seves condicions són:

| Productos | Plan Spark<br>Límites generosos para aficionados<br>Sin cargo | Plan Flame<br>Precios predecibles para apps en expansión<br>USD 25/mes | Plan Blaze<br>Calcula los precios de las apps a gran escala<br>Pago por uso<br>✓ Free usage from Spark plan included* |
|---|---|---|---|
| **Incluido sin cargo**<br>A/B Testing, Analytics, App Indexing, Authentication (except Phone Auth), Cloud Messaging (FCM), Crashlytics, Dynamic Links, Invites, Performance Monitoring, Predictions, and Remote Config. | ✓ Incluidos | ✓ Incluidos | ✓ Incluidos |
| **Cloud Firestore** | | | |
| Stored data | 1 GB total | 2.5 GB total | $0.18/GB |
| Bandwidth | 10GB/month | 20GB/month | Google Cloud Pricing |
| Document writes | 20,000/día | 100K/day | $0.18/100K |
| Document reads | 50,000/día | 250,000/día | $0.06/100K |
| Document deletes | 20,000/día | 100K/day | $0.02/100K |
| **Storage** | | | |
| GB almacenados | 5 GB | 50 GB | USD 0.026/GB |
| GB descargados | 1 GB/día | 50 GB/día | USD 0.12/GB |
| Operaciones de carga | 20,000/día | 100K/day | $0.05/10k |
| Operaciones de descarga | 50,000/día | 250,000/día | $0.004/10k |

Figura 33: Costos produits per l'ús de Cloud Firestore i en un futur del Storage(per guadar imatges o altres)[28]

Si l'aplicació arribés a ser usada per una gran nombre d'usuaris s'hauria de calcular quina quantitat d'informació genera cada usuari i tractar de minimitzar al màxim la informació i les interaccions amb la base de dades de Cloud Firestore. Tot i això, si fos impossible reduir l'ús al mínim per no tenir costos s'hauria de plantejar assumir els costos del *Plan Flame* i oferir als usuaris que realitzessin donacions. Si finalment no hi hagués més opcions perquè l'aplicació agafés molta volada s'haurien d'assumir els costos del *Plan Blaze* i buscar noves formes per finançar el projecte. Aquesta última part es veu llunyana, és per això que no s'ha realitzat encara cap planificació a nivell de costos més extensiva.

## 6.15 Xat

Finalment ha donat temps d'implementar el xat, una part del codi que emmirallant-se en el tutorial *Firebase Firestore xat App- Kotlin Android Tutorial* [29] [30] i gràcies a l'estructura del codi del present projecte, ha donat un gir de 180º a la

forma de compartir informació dintre d'un projecte entre usuaris.

S'ha desenvolupat una línia de xat semblant al tutorial però adaptada al codi del projecte, algunes classes que s'oferien s'han fet servir, sobretot les referents al RecyclerView i l'adaptador dels ítems. En canvi s'han usat les classes pròpies per als mètodes relacionats amb la interacció amb la base de dades, *MainInteractor*, i s'ha creat una nova classe per al fragment que permet veure el xat *InProjectFragment* i les tasques dels projectes.

Està clar que el xat es troba en una fase molt verge però dona llum a la cooperació dins dels projectes i li imprimeix un caràcter molt més interactiu a l'aplicació. No es valorarà doncs tampoc el disseny del mateix, només cal seguir en aquesta direcció i afegir certes funcionalitats bàsiques al xat i altres pròpies de l'aplicació. Alguns eemples de futures implementacions serien: veure els missatges que et queden pendents i no has llegit; que al xat es puguin veure les tasques que es creen i qui ho ha fet; etc.

## 6.16 Llengües

Gràcies a l'opció que ofereix el propi Android Studio per editar el fitxer *string.xml*, que hauria de contenir tots els textos que es fan servir a l'app, s'han pogut crear els que estan relacionats amb les altres llengües. Mentre es creaven els *Strings* que es necessitaven per a ús genèric en anglès també es creaven els de les altres dues llengües que ofereix l'aplicació: català i castellà.

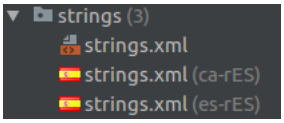


Figura 34: Arxius per les respectives llengües

Autòmaticament l'aplicació detecta el llenguatge del dispositiu i en cas de que estigui en català o castellà fa servir els *Strings* respectius i en cas contrari usa la versió genèrica en anglès dels mateixos.

# 7 UI: Proves i resultats

La User Interface, interfície d'usuari, d'aquesta app és una interfície gràfica i per desenvolupar-la s'ha seguit un procés creatiu més enllà d'anar afegint a l'atzar els diferents elements gràfics i seguir la guia de disseny que ofereix *material.io*. És per això que a part de fixar-se en aplicacions conegudes, com s'ha comentat anteriorment, hi ha parts del disseny completament noves respecte altres aplicacions: tant les que implementen funcionalitats semblants a la present aplicació com respecte d'altres més *mainstream*.

El disseny s'ha anat comentant junt amb les explicacions del desenvolupament de les diferents funcionalitats. Algunes parts del disseny són el fruit d'hores de disseny sobre el paper, com els que s'aprecien a l'apèndix F, i d'estar-se assegut pensant en la millor solució per mostrar la informació de la forma menys sobrecarregada i més agradable a l'usuari. A part, tot i que no hi ha una estil de disseny molt definit, si que s'ha escollit una línia per utilitzar certs colors clars que no sobrecarreguessin el disseny i alhora imprimissin en l'usuari una estètica de colors diferenciada d'altres apps. D'altra banda s'ha intentat que l'usuari tingués una forma fàcil de diferenciar projectes per colors i que alhora es pugessin afegir categories que en el cas de les tasques definirien el color de les mateixes. A més, s'han reutilitzat les mateixes icones per a referir-se als mateixos elements.

Els tests clarament valoraran el disseny però en cap cas valoren el codi. Inicialment es vol aclarir que aquest test compleix una funció concreta que és comprovar la usabilitat de la UI de l'aplicació i ajudar així a valorar les seves potencialitats i mancances d'una forma relativament objectiva. Per aconseguir aquest propòsit s'han agafat diferents usuaris amb diferents qualitats i sense cap explicació ni guiatge previ se'ls ha empès a realitzar diferents tasques.

## 7.1 Proves

Finalment s'han aconseguit diferents usuaris per a realitzar els tests amb característiques diferents:

- **Laura:** 50 anys. Poc acostumada a usar la tecnologia, de la generació que va créixer sense aquesta. Utilitza Instagram i WhatsApp, però té dificultats. Usuària d'Android.
- **Ricardo:** 50 anys. Més acostumat a fer servir tecnologia a nivell d'usuari però de la generació que va créixer sense la mateixa. Usa diverses aplicacions. Usuari d'IOS.
- **Marta:** 19 anys. De la generació que tot i no créixer amb les xarxes socials va començar a usar-les durant l'adolescència. Utilitza Instagram, WhatsApp i altres aplicacions de xarxes socials . Usuària d'IOS.

- **Òscar:** 12 anys. De la generació que des de petit ha crescut usant la tecnologia tant per a les xarxes socials com a nivell de consoles. Usuari d'IOS.

Primer de tot testejaran l'aplicació a partir d'un seguit de passos escollits per probar diferents funcionalitats:

| Tasques | |
|---|---|
| 1 | Autentificar-se |
| 2 | Crear una una rutina setmanal per als divendres amb una categoria |
| 3 | Crear un projecte |
| 4 | Crear una tasca concreta, sense repetició, i associar-la al projecte |
| 5 | Mirar al calendari el proper dia que toca realitzar la rutina |
| 6 | Afegir un altre usuari al projecte i enviar un missatge al del projecte |

Taula 1: Tabla tareas.

| Tasques | Laura | Ricardo | Marta | Òscar |
|---|---|---|---|---|
| 1 | ✓ | ✓ | ✓ | ✓ |
| 2 | ✓ | ✓ | ✓ | ✓ |
| 3 | ✓ | ✓ | ✓ | ✓ |
| 4 | ✓ | ✓ | ✓ | ✓ |
| 5 | ✓ | ✓ | ✓ | ✓ |
| 6 | X | ✓ | ✓ | ✓ |

Taula 2: Tabla resultados.

Hi ha dos mòbils principals que han portat a definir el test de funcionalitats de l'anterior manera. El primer i principal motiu era que els usuaris provessin les diferents eines que donava l'aplicació i es pogués apreciar la seva usabilitat a curt termini. L'altre era visualitzar possibles errors o problemàtiques més genèriques que es poguessin trobar els usuaris depenent del seu coneixement sobre les tecnologies. Inicialment es va dir que hi havia la intenció de provar l'ús de l'aplicació a mitja termini, dues setmanes aproximadament, però al estar en una versió inestable, es va valorar que era millor perfeccionar-la i testejar totes les funcionalitats que no deixar funcionalitats a mig fer i donar massa importància al test a mitjà termini. Tot i això, durant més de dues setmanes s'ha estat usant de forma funcional l'aplicació per organitzar-se, tant per fer el TFG com per la vida quotidiana. .

En segon lloc es realitzarà una enquesta on es realitzaran diferents preguntes al voltant del disseny de l'aplicació, la funcionalitat i la usabilitat. Es realitzaran les següents preguntes on es demanarà una valoració del 1-10 en algunes i a part es realitzaran algunes preguntes de resposta oberta:

- 1.- Ha estat una experiència agradable?

- 2.- Com valoraries els colors que utilitza l'aplicació?
- 3.- Valores positivament el disseny de l'aplicació(botons, formes d'introduir text, etc)?És a dir, no se t'ha fet pesat?
- 4.- T'ha semblat útil el disseny de la pantalla inicial?
- 5.- I el de la pantalla de projectes?
- 6.- I el calendari?
- 7.- T'ha semblat fàcil la introducció de tasques, rutines o projectes?
- 8.- Milloraries alguna cosa de les pantalles?(oberta)
- 9.- Milloraries alguna cosa d'aquests diàlegs?(oberta)
- 10.- D'entre les diferents possibilitats que ofereix quina destacaries?(oberta)
- 11.- D'entre les diferents possibilitats que ofereix quina milloraries?(oberta)

| Preguntes | Laura | Ricardo | Marta | Òscar |
|---|---|---|---|---|
| 1 | 8 | 9 | 8 | 7 |
| 2 | 8 | 10 | 7 | 10 |
| 3 | 7 | 9 | 6 | 7 |
| 4 | 8 | 9 | 10 | 8 |
| 5 | 7 | 8 | 10 | 8 |
| 6 | 10 | 10 | 8 | 7 |
| 7 | 9 | 10 | 10 | 6 |

Taula 3: Tabla resultados.

A part tenim a l'annex D les respostes a les preguntes obertes que ens donen informació molt valuosa sobre l'opinió de l'aplicació.

## 7.2 Resultats

Es pot apreciar com les majors dificultats estan associades a la vista dels projectes, l'adició d'usuaris als projectes i l'edició del mateix projecte. Tot i això, en general la valoració és molt positiva i no hi ha cap crítica a nivell fonamental que faci trontollar el projecte. Han accentuat la utilitat de l'aplicació i les possibilitats que ofereix. A part, han tingut molt bona rebuda les funcionalitats relacionades amb compartir tasques i projectes, de forma innovadora, que ofereix l'aplicació. Per tant, es pot dir que en gran part els usuaris han valorat molt positivament l'app tant a nivell d'ús personal com a nivell grupal.

Així doncs, després de veure com se'ls ha fet entretingut i simple el seu ús és

important subratllar que alguns dels usuaris que han testejat l'aplicació estaven poc familiaritzats amb la tecnologia i havien tingut problemes anteriorment al usar altres aplicacions com WhatsApp o Instagram. És per això que els diferents tests porten a pensar que s'ha escollit un disseny força adequat per a tots els públics. Tot i això, després de veure alguns petits errors i millores que els propis usuaris veien a primera vista es podia apreciar clarament que es tracta d'una versió inacabada. Alhora al dur a terme aquestes proves amb usuaris reals s'ha decidit que abans de treure l'aplicació al públic haurà de passar per alguns tests concrets i un ús mitjanament prolongat, entre una i dues setmanes, per diversos usuaris i absorbir totes les crítiques possibles per poder millorar l'app. Tots aquests tests han fet sorgir la idea d'oferir una forma propera i innovadora de rebre suggeriments per part dels usuaris i facilitar una comunicació bidireccional amb els desenvolupadors.

Es valora molt positivament haver realitzat aquests tests i donen informació crucial tant sobre les idees bones o dolentes més generals lligades a l'aplicació com sobre elements concrets del disseny o errors que no s'havien plantejat. També és important afegir que durant les últimes setmanes, que es va utilitzar l'aplicació per organitzar-se tant pel TFG com per la vida quotidiana, s'ha trobat molt útil l'aplicació. Tot i això, caldria que algunes persones, que no fossin el propi creador, provessin l'aplicació a mitjà termini quan aquesta estigués en una versió més avançada.

# 8 Conclusions

Per acabar, després d'haver exposat el procés que s'ha seguit per tenir aquesta versió *"beta"* s'han extret diferents valoracions. Segons la gran divisió que s'ha vist durant tota l'explicació entre codi i disseny es passarà a valorar aquestes dues parts respectivament.

Per una banda hi ha el codi, utilitzant certes tecnologies, que des de la prespectiva de l'usuari no es poden apreciar normalment. En aquest projecte s'ha invertit inicialment força temps en escollir les tecnologies per tal d'obtenir una opció innovadora però que fos robusta. Després d'haver implementat el projecte i veient com han anat evolucionant respectivament es pot destacar que han estat tries molt correctes.

Primer es vol destacar que el llenguatge Kotlin, que ha ofert noves possibilitats i ha estat una experiència molt agradable. Tot i això, al no estar familiaritzats al 100% amb el llenguatge el codi encara es pot optimitzar més fent servir algunes funcionalitats natives de Kotlin que no es troben en java. A part, Kotlin/Native cada vegada està més proper a tenir una versió funcional, 1.0 o major, que permeti programar l'aplicació multiplataforma robustament. És veritat que Flutter ha guanyat importància i Reactive Native també, però hi ha un element que fa que Kotlin/Native, com passa amb els projectes de java, destaqui per sobre de les altres opcions que és la interoperabilitat que ofereix entre llenguatges. Actualment ja es pot interoperar tant amb java com amb Objective-C o d'altres llenguatges.[31] [32] [33] [34] Tot i això, no se li està donant gaire publicitat però està tenint una molt bona acollida per la comunitat de programadors d'Android, o provinents de Java en general, que veuen en Kotlin un llenguatge que, com ja s'ha explicat, millora certs problemes de Java i implementa noves possibilitats. Ha estat molt útil i fàcil aquest nou llengutage.

Després, haver escollit Cloud Firestore ha estat una molt bona elecció, l'evolució d'aquesta tecnologia i totes les opcions que dona als projectes creats amb Firestore no només són cada vegada més estables sinó que alhora està augmentant diàriament les eines que ofereix. Permet tractar la base de dades de diferents maneres, també permet definir Cloud Functions que permeten executar certes funcions pròpies dintre de la base de dades quan des de l'aplicació es realitzessin certes interaccions amb la base de dades i un seguit de funcions sobre les que no s'estendrà l'explicació.

A part de les tecnologies que utilitza també ha estat molt important l'estructura del codi, MVP+Interactor. Va costar força temps buscar la informació i tenir el projecte completament definit a nivell de classes i carpetes. Un cop ja s'havia escollit i es va començar a usar s'ha valorat molt positivament. Varies vegades s'han vist les facilitats que oferia per implementar certes funcionalitats com s'explicava a alguns tutorials però adaptat a la present aplicació. Per contra, és important reconèixer que hi ha alguns elements que es poden millorar i fent servir totes les eines que brinda Kotlin es podrien reestructurar algunes parts i fer més eficients altres. En línies generals però, no caldria modificar cap part i per a posteriors aplicacions

amb base de dades i models, es força recomanable aquesta estructura.

El codi està en un repositori privat de Github també i en un futur es podrien afegir fàcilment col·laboradors i gestionar els canvis que realitzessin al projecte amb Github.

D'altra banda es valora força positivament el disseny tant a nivell de divisió de pantalles, com les formes d'introduir informació com editar els plans que ja han creat els usuaris. Està clar que s'haurien de millorar alguns elements per què hi hagués més maneres d'accedir-hi des de diferents parts de l'aplicació. Com per exemple que es pogués accedir des d'altres llocs al xat o poder modificar els projectes o tasques d'altres maneres, etc. També s'haurien de valorar amb més profunditat els colors que s'usaran perquè si no es realitza una bona elecció pot tirar enrere als usuaris. A part, alguns elements s'han programat per a què complissin la funcionalitat desitjada sense haver aprofundit molt en els motius del disseny.Tot i això, en les vistes principals s'ha pensat força a l'hora de decidir quina era la millor manera de mostrar els plans a l'usuari. En general després d'haver-la fet servir i de realitzar els tests sembla que els dissenys per les tres pantalles principals: *TO-DO*, *Projectes* i *Calendari*; han estat un encert. Les mancances principals estan associades a la vista dels projectes on es pot escollir o veure les tasques pendents del projecte o el xat. Aquesta vista ha estat l'última i es troba encara en un estat molt verge, però com la base ja està implementada només seria necessari perfeccionar el que hi ha actualment.

Es valora molt positivament haver realitzat un projecte autònom on s'havia d'aconseguir desenvolupar des del no res un objectiu bastant elaborat. Ha estat un procés complex on es barrejaven forces sentiments i amb èpoques més àlgides i altres més difícils. En general, però, en aquests últims mesos no hi ha hagut treva. S'ha treballat diàriament, primer amb l'aplicació i després amb la memòria. Malgrat ha estat un camí llarg s'ha gaudit molt i en cap moment s'ha fet pesat. Hi ha hagut parts més entretingudes que altres però si s'ha de destacar alguna, la millor part han estat els tests. Ha estat l'apartat més agradable al veure-s'hi materialitzat tot el treball en el gaudi de les persones que l'han fet servir i poder apreciar amb una prespectiva *neutre* els errors i encerts de l'aplicació.

Un cop s'ha arribat a aquest punt si es mira enrere es pot veure tot el treball realitzat i per tant diferenciar allò que s'ha fet correcte o incorrectament. És difícil preveure tots els factors d'un projecte d'aquesta envergadura però tenint en compte la planificació prèvia es pot afirmar que s'han complert els objectius del projecte i, fins i tot, ha donat temps d'implementar aquelles parts que es preveien com a opcionals. Així doncs, tot i que hi ha alguns apartats que es poden millorar s'està satisfet amb la versió a la que s'ha arribat al acabar aquest projecte. Finalment, després d'un llarg camí es pot dir que ha estat una gran experiència que deixa una empremta inesborrable i empeny seguir endavant amb el projecte i no defallir.

## 8.1 Treball futur

Com afegit es vol destacar com s'albiren diferents línies de treball que defineixen el futur de l'aplicació. Un cop arribats a la situació actual es pot veure com el que distingirà l'aplicació d'altres seran: per un cantó els projectes cooperatius i la comunicació entre usuaris; i per l'altre el disseny de l'aplicació. A part, també serà necessari aprofundir en el tractament d'errors i el llançament al mercat. Així doncs, en les línies d'aquests quatre eixos s'explicarà una mica més cap a on es dirigeix l'aplicació:

- **Projectes i cooperació:** S'haurà de millorar la vista que s'obre al prémer sobre els projectes. També es millorarà el xat per fer-lo més agradable: afegint opcions per penjar fotografies, documents o altres; deixant veure tasques o rutines en el mateix xat; permetent afegir usuaris més fàcilment; definint permisos per limitar l'edició del projecte; en general millorar el disseny del xat; etc. Alhora s'afegirà la possibilitat de tenir amics i una nova forma per incloure'ls als projectes. També es permetrà xatejar dintre d'una tasca. Es podrà compartir les tasques amb el calendari de Google o altres. Serà més fàcil convidar persones a l'aplicació i compartir els projectes en altres xarxes socials.

- **Disseny de l'aplicació:** podrien córrer moltes línies per explicar cadascun dels elements que es volen millorar però el que s'ha vist més útil per valorar-lo i obtenir noves idees és realitzar tests amb usuaris. S'ha pogut apreciar amb els tests realitzats fins ara que s'han de facilitar forces opcions i s'ha de valorar els colors generals que usa l'aplicació. Sobretot no s'ha de donar res per suposat ni s'ha de creure que això o allò no és tan difícil per l'usuari. S'ha vist com hi ha certes funcionalitats de l'aplicació en les que és gairebé necessari emmirallar-se en altres aplicacions perquè els usuaris ja tenen apreses aquestes formes d'interactuar. Hauria d'existir al menú una opció d'ajuda on s'expliquessin les diferents funcionalitats de l'aplicació i com es poden fer servir. També podria servir per penjar-hi vídeos explicatius de noves funcionalitats incloses a l'aplicació.

- **Errors:** S'ha de facilitar a l'usuari poder veure quan s'equivoca i oferir-li possibilitats. No s'ha de creure mai que l'usuari deduirà on està l'error.

- **Mercat:** Quan es tingui una versió estable que hagi passat per diferents iteracions i tests amb usuaris s'hauria de valorar quina és la millor opció de llançar-la al mercat i donar-li el màxim de publicitat. Adequant-se a les possibilitats econòmiques s'hauria de buscar la millor manera per donar a conèixer l'aplicació al públic.

Es podria seguir parlant del futur i de com i què es faria, però probablement no s'acabaria mai. Ara només queda seguir en aquesta direcció i no desviar-se del camí recorregut.

# Referències


[1] Dr. Ricardo E. Reolid-Martínez; Dra. María Flores-Copete; Dra. Mónica López-García; Dra. Pilar Alcantud-Lozano; Dra. M. Candelaria Ayuso-Raya; y Dr. Francisco Escobar-RabadánaBatut: *Frecuencia y características de uso de Internet por adolescentes españoles. Un estudio transversal*, 2016 `http://www.sap.org.ar/docs/publicaciones/archivosarg/2016/v114n1a03.pdf`

[2] Kent Beck et al.: *Manifesto for Agile Software Development*, 2001 `http://agilemanifesto.org/iso/ca/principles.html`

[3] Castillo, Toni. *Las palabras de Zuckerberg no solucionan el mayor problema de Facebook: la transparencia.* genbeta.com, 22/03/2018 Recuperat de https://www.genbeta.com/redes-sociales-y-comunidades/las-palabras-de-zuckerberg-no-solucionan-el-mayor-problema-de-facebook-la-transparencia

[4] Badia, Quique. *César Rendueles: 'Airbnb o Uber es venen com fenòmens contraculturals fruit de l'energia cooperativa'*, eltemps.cat, 27/02/2018. Recuperat de https://www.eltemps.cat/article/3458/cesar-rendueles

[5] Badia, Quique. *Pablo Capilla: 'La reflexió sobre el periodisme, com més promíscua millor'*, eltemps.cat, 11/01/2018. Recuperat de https://www.eltemps.cat/article/3016/pablo-capilla-reflexio-periodisme-promiscua

[6] Soto Ivars, Juan: *Arden las redes*,1a edició, Editorial Debate, 2017

[7] Gràfica del Maig 2017 - Maig 2018, gs.statcounter.com. Basada en més de 10 mil milions de pàgines vistes al mes registrades en més de 2 milions de llocs web. Recuperat de http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201705-201805-bar

[8] Cruxlab, Inc, *Xamarin vs Ionic vs React Native: differences under the hood*, cruxlab.com, 20/09/2017. Recuperat de https://cruxlab.com/blog/reactnative-vs-xamarin/

[9] Team Novoda, *React Native, Flutter, Xamarin: a comparison*, blog.novoda.com, 25/01/2018. Recuperat de https://blog.novoda.com/react-native-flutter-xamarin-a-comparison/

[10] Shafirov, Maxim: *Kotlin on Android. Now official*, blog.jetbrains.com, 17/05/2017. Recuperat de https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/

[11] Breslav, Andrey: *KotlinConf 2017 - Deep Dive into Kotlin/Native by Andrey Breslav*, JetBrainsTV , 15/11/2017. Recuperat de https://www.youtube.com/watch?v=3Lqiupxo4CE

[12] Google Developers: *Cómo elegir tu base de datos: Cloud Firestore o Realtime Database*, firebase.google.com, 10/05/2018. Recuperat de https://firebase.google.com/docs/database/rtdb-vs-firestore

[13] Leiva, Antonio: *MVP for Android: how to organize the presentation layer*, antonioleiva.com, 16/04/2014. Recuperat de https://antonioleiva.com/mvp-android/

[14] Ali, Janishar: *Android MVP Architecture Extension with Interactors and Repositories*, blog.mindorks.com, 29/12/2017. Recuperat de https://blog.mindorks.com/android-mvp-architecture-extension-with-interactors-and-repositories-bd4b51972339

[15] Font: https://developer.android.com/reference/android/app/Activity

[16] Font: https://developer.android.com/guide/components/fragments

[17] Fonts: https://developer.android.com/reference/android/app/DialogFragment and https://developer.android.com/reference/android/support/v4/app/DialogFragment

[18] Projecte de la llibreria Groupie: https://github.com/lisawray/groupie

[19] Projecte de la llibreria MaterialDrawer per al menú: https://github.com/mikepenz/MaterialDrawer

[20] Compte que ha desenvolupat aquesta versió del FloatingActionButton: https://github.com/Clans. Recuperat de https://github.com/Clans/FloatingActionButton

[21] Projecte Ambilwarna que desenvolupa un colorPicker dialog per escollir un color: https://github.com/yukuku/ambilwarna

[22] Google Project FlexBoxLayout: https://github.com/google/flexbox-layout

[23] Projecte CircleImageView per Henning Dodenhof:https://github.com/hdodenhof/CircleImageView

[24] Permisos per a fer captures de pantalla de productes de Google: https://www.google.com/permissions/using-product-graphics.html

[25] By HabitRPG, Inc. [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons. Recuperat de https://commons.wikimedia.org/wiki/File:Habitica_Android_Screenshot.png

[26] Guia de disseny. Recuperat de https://material.io/design/.

[27] Font: https://developer.android.com/reference/android/support/design/widget/BottomNavigationView

[28] Font: https://firebase.google.com/pricing/

[29] Reso Coder: Firebase Firestore Chat App: Creating a xat Channel (Ep 4) - Kotlin Android Tutorial, resocoder.com, 17, 4, 2018. Recuperat de https://www.youtube.com/watch?v=cG26G8WNv-0

[30] Reso Coder: Firebase Firestore Chat App: Send Text Messages (Ep 5) - Kotlin Android Tutorial, resocoder.com, 27, 4, 2018. Recuperat de https://www.youtube.com/watch?v=ybS6epU1NGQ

[31] Jetbrains: *Calling Java code from Kotlin*, kotlinlang.org. Recuperat de https://kotlinlang.org/docs/reference/java-interop.html

[32] Jetbrains: *Calling Kotlin from Java*, kotlinlang.org. Recuperat de https://kotlinlang.org/docs/reference/java-to-kotlin-interop.html

[33] Igotti, Nikolay: *Kotlin/Native v0.4 released: Objective-C interop, WebAssembly and more*, blog.jetbrains.com, 16,11, 2017. Recuperat de https://blog.jetbrains.com/kotlin/2017/11/kotlinnative-v0-4-released-objective-c-interop-webassembly-and-more/

[34] Igotti, Nikolay: *Kotlin/Native v0.5 released: calling Kotlin from Swift and C, LLVM 5 and more*, blog.jetbrains.com, 19,12, 2017. Recuperat de https://blog.jetbrains.com/kotlin/2017/12/kotlinnative-v0-5-released-calling-kotlin-from-swift-and-c-llvm-5-and-more/

# Appendices

## A Casos d'ús de l'usuari no identificat

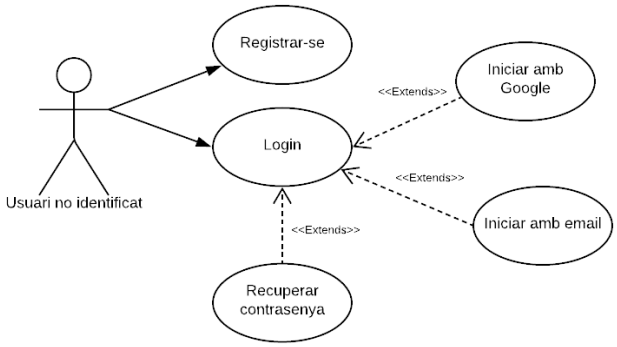


Figura 35: Casos d'ús de l'usuari no identificat

## B Casos d'ús de l'usuari identificat

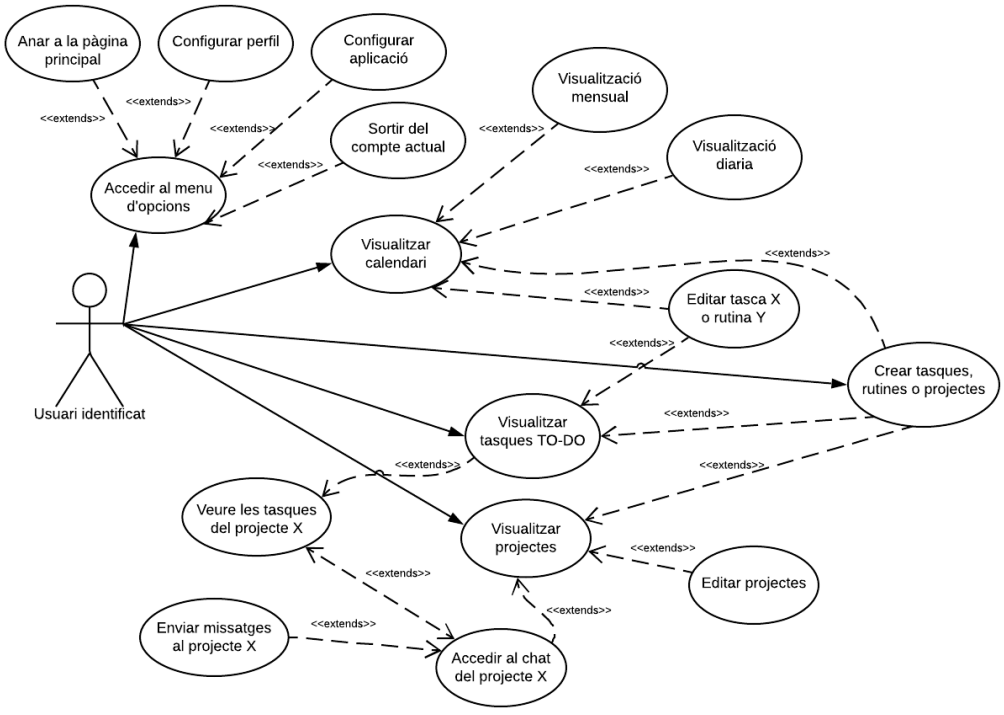


Figura 36: Casos d'ús de l'usuari identificat

# C Diagrama de flux de la creació d'una tasca

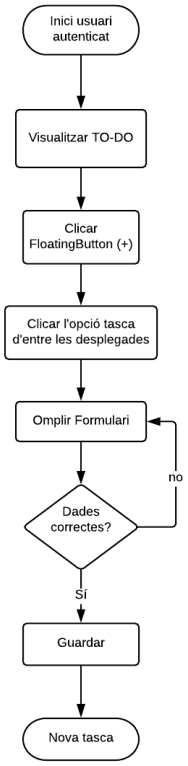


Figura 37: Diagrama de flux de la creació d'una tasca

# D Test: respostes preguntes obertes

(Es tracta d'una transcripció literal) - Laura: (8.-)De proyectos, para editar se deberia poder editar dentro de la vista del proyecto. Añadir usuario deberia estar más fàcil, dentro del mismo proyecto. Error en la fecha del proyecto o tarea si añades el mismo dia per al cabo de un año te sale el mismo año. Le gustaria que le salieran algunos colores ya dados. (9.-)Le gustaria que se puedieran editar las categorías.(10.-) Destaca el poder tener proyectos y te ayuda a organizarte. Te empuja a pensar en hacer otros proyectos y te permite organizar tu vida y te da ganas de hacer cosas y planificarte. (11.-) Poder conectarlo con el colegio, mail y otras aplicaciones. Estaria bien que mostrara de alguna forma cuando el usuario introduce datos de forma incorrecta.

- Ricardo: (8.-)Todo bastante correcto. La limitacion de carácteres le gustaria a ampliarlo. En general correcto, pero en los colores le pareceria practico la paleta de colores ofreciera ya unos colores basicos.(9.-) Le parece correcto. (10.-) Le gusta mucho lo de poder compartir proyectos con otras personas y poder mandarles mensajes. Te ayuda organizarte en los proyectos personales y tenerlo todo

calendarizado.(11.-) Que al añadir usuario se puediera ir a la agenda personal, tener amigos u otros. Poder añadir tareas o proyectos al google calendar. Avisos del xat. Que mandara notificaciones al mail.

Marta: (8.-) En el calendario, que fuera más visual el tema del més. Más colorido y no solo uno y cero y dos colores. Valora positivament la pantalla de proyectos. Se deberian poner mensajes en cada tarea, que tuvieran un xat único. Poner apartado de consejos donde esten los nuevos elementos implementados y tambien antiguos. Así puedes ver fácilmente como funciona. (9.-) Así les gusta. Aunque se podria poner una descripción de la tarea. (10.-) Le ha gustado que puedas poner rutinas. Le parece muy útil lo de proyectos para hacer trabajos, para organizar viajes, etc. (11.-) Se podria poner algo de retos, frases para motivar, sugerencias...; para motivar a la persona. Para que cuando realices una tarea te salgan frases motivadoras, al crear la tarea que el usuario se coloque las frases que le saldran motivandolo para cuando la este haciendo.

Òscar: (8.-) Del projecte no canviaria res. Del calendari faria que lo del Mes estigués en una única pantalla. (9.-) Que al diàleg es vegi més gran lo d'assignar projectes. Que lo de categories es posés directament al selector la categoria seleccionada. Més amplis els diferents desplegables tant de categories com de projectes.(10.-) Li agradat que es pugui afegir fàcilment a un projecte les tasques o rutines ja creades. Valora positivament la pantalla inicial perquè et permet veure totes les tasques que et queden per fer.(11.-) Poder triar els colors de l'aplicació. Que es poguessin invertir els colors de l'aplicació. Ficar en una pàgina directament Mes i Dia de tal forma que es pogués veure directament.

# E Dissenys en paper

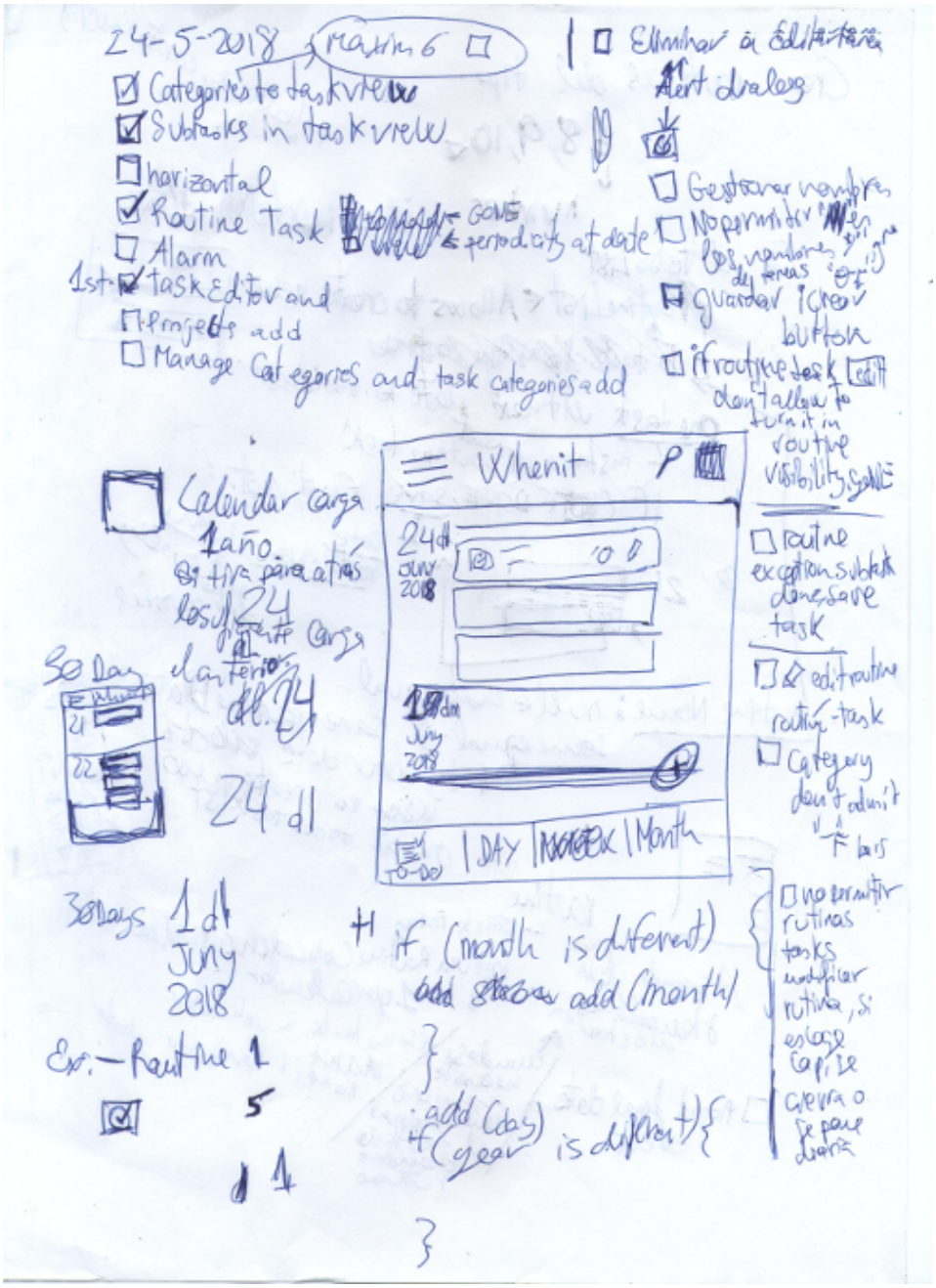


Figura 38: Dissenys varis en paper

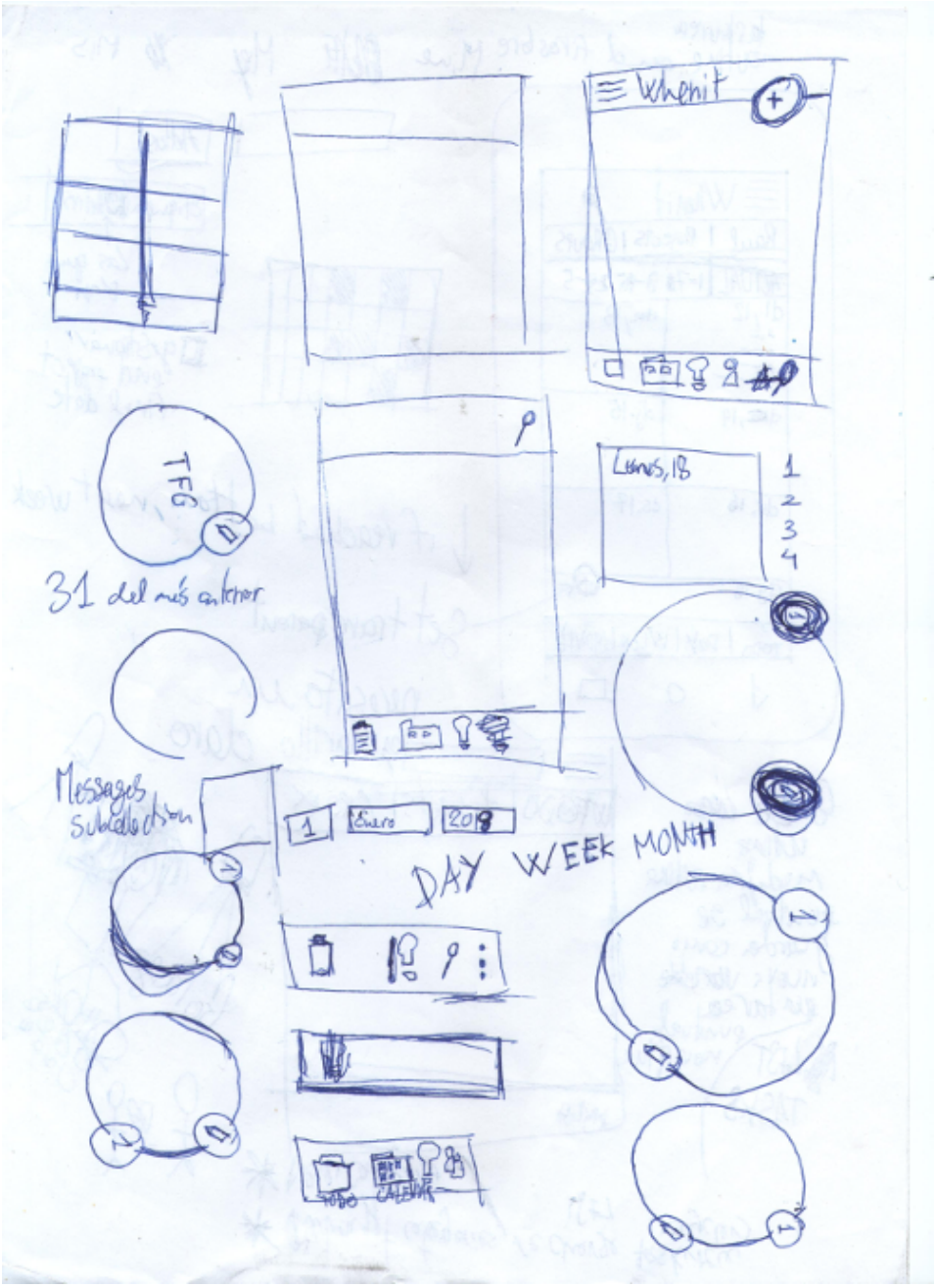


Figura 39: Dissenys varis en paper

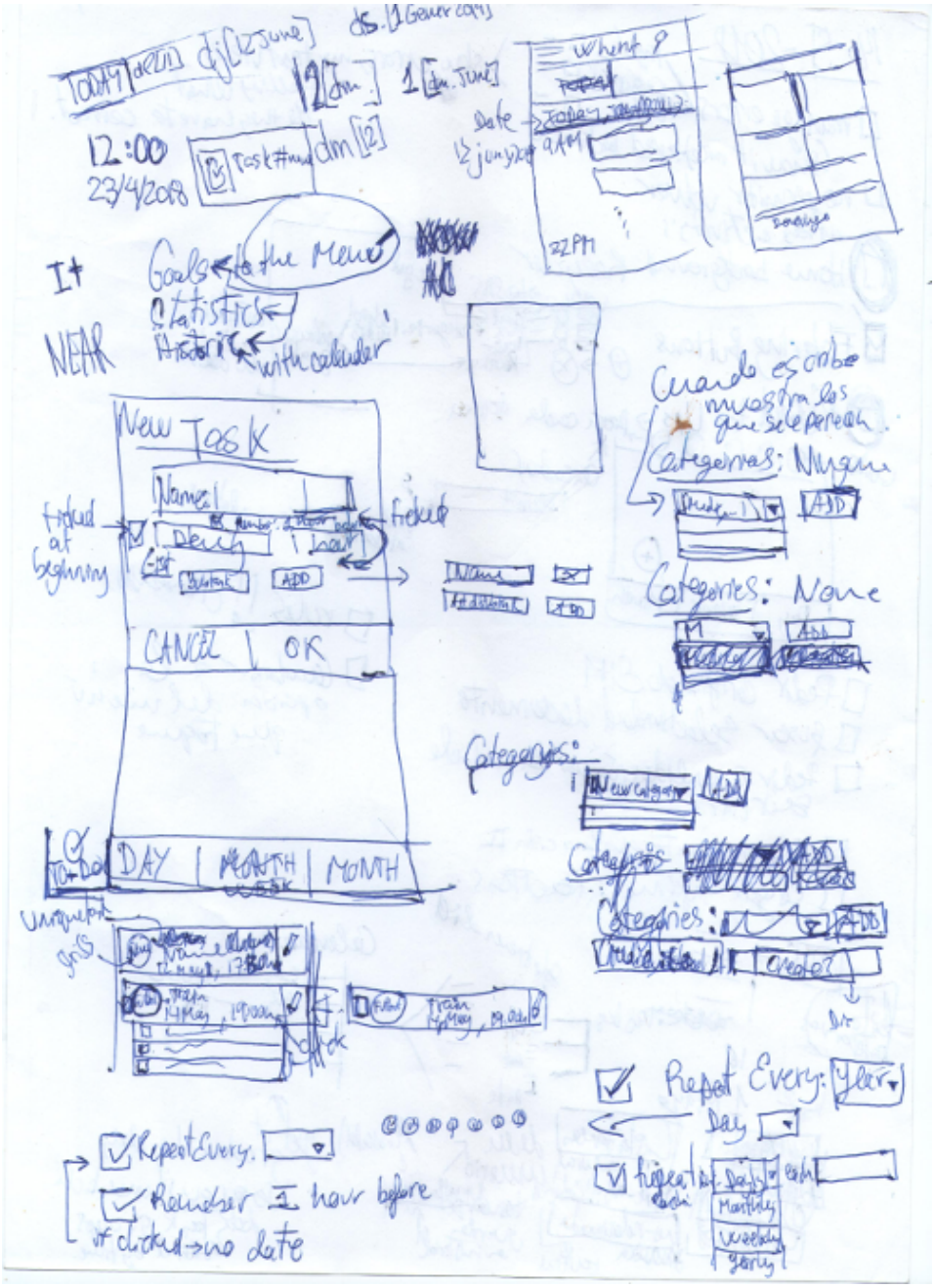


Figura 40: Dissenys varis en paper

# F Nou nom: Whenit

Temporalment s'ha decidit reformular el nom a Whenit perquè es va trobar que ja existia una aplicació que es deia Weplan.